**Profesionālās izglītības kompetences centrs**

**„RĪGAS VALSTS TEHNIKUMS”**

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

**SIA “Pirmā programmēšanas skola” Mācību sistēmas izveide ar automatizētu uzdevumu risināšanu**

Mājaslapas izveide

Audzēknis: L.Liepenieks

Vadītājs I.Litvjakovs

Normu kontrole O.Sabanska

**Rīga**

**2021**

# ANOTĀCIJA

Kvalifikācijas darba ietvaros izstrādāta mācību sistēma, priekš uzņēmuma SIA “Pirmā programmēšnaas skola”. Projekta mērķis bija izveidot automatizētu uzdevumu sistēmu, lai skolēni varētu pārliecināties, par savu risinājumu korektumu. Galvenais sistēmas uzdevums ir iespēja jebkuram reģistrētam lietotājam spēt iesūtīt jebkuru uzdevumu un apskatīties vai visi uzdevumā pievienotie testi ir izpildīti korekti. Sistēmas ietvaros tika izveidota gan šī galvenā funkcionalitāte, no lietotāja puses, gan iespēja administratoriem pārvaldīt sistēmu – pievienojot, dzēšot un labojot dažādu mājaslapai nepieciešamo informācjiu. Darbs tika izstrādāts pamatā izmantojot uz PHP valodas balstīto Laravel ietvaru.

Kvalifikācijas darba skaidrojošais raksts sevī ietver ievadu, kas kalpo kā skaidrojums un pamatojums, kāpēc tika veikta sistēmas izstrāde. Uzdevuma nostādni, kas kalpo kā skaidrojums vēlamajam rezultātam sistēmas izveidē,. Prasību specifikācija, kas sastāv no trīs daļām – ieejas un izejas informācijas apraksts, funkcionālās prasības, nefukncionālās prasības – šī nodaļa raksturo konceptuālos mērķus un informāciju, kas nepieciešama programmas korektai darbībai. Uzdevuma risināšanas līdzekļu pamatojums, kas raksturo sistēmas izstrādē izmantotās tehnoloģijas un sistēmas darbībai nepieciešamās apakšsistēmas un spraudņus. Programmatūras produkta modelēšana un projektēšana, kas skaidro sistēmas vēlamo rezultāta struktūru un datu plūsmu. Datu struktūras apraksts, kas skaidro datubāžu struktūru un korekto specifikāciju, mājaslapas korektai darbībai. Lietotāja ceļvedis, kas izskaidro sistēmai nepieciešamās tehniskās un programmatūras prasības, kā arī individuāli skaidro, katra loga un skata saistošo jēgu lietotājam.

Kvalifikācijas darba skaidrojošais raksts sastāv no 74 lappusēm, 53 attēliem, 3 pielikumiem un 10 tabulām

# ANNOTATION

As a result of the qualification work there has been developed a automatized exercise management system for SIA”Pirmā programmēšanas skola”. Purpose of the project was to create an automatized exercise grading system, so that students could test their code, and as a result see if it has been correct. Main aim of the system was that any registered user could send any public exercise and see if their solution has been correct – all tests have been passed. As a result of development this functionallity has been implemented, from users side, also there has been developed administrators side, from which admin users can manage any and all information which is essential for the program to work correctly. Project was developed mainly within Laravel framework, which is based on PHP server side language.

This documentation within itself holds introduction statement, which serves as explanation and substantination, why did this system was developed. Task statement, which serves as explanation for the expected result of development. Task formulation, which consists of three parts – description of input and output information, functional requirements, non-functional requirements – this chapter describes conceptual target’s and information, which would be essential for systems correct behaviour. Projects development technical means explanation, which describes what technologies were used in systems development and plugins which are essential for correct behaviour. Systems product’s modeling and projecting, Description of data structures, which describes database structure and correct specification for all data to be validated and used correctly. Users guide, which explains system requirements -both software and hardware – setup, and explanation of every window and explanation how and why it should be used.

Qualification project consists of 74 pages, 53 images, 3 attachments and 10 tables.

# SATURS

[IEVADS 5](#_Toc74045212)

[1.UZDEVUMA NOSTĀDNE 6](#_Toc74045213)

[2. PRASĪBU SPECIFIKĀCIJA 7](#_Toc74045214)

[2.1. Ieejas un izejas informācijas apraksts 7](#_Toc74045215)

[2.1.1. Ieejas informācijas apraksts 7](#_Toc74045216)

[2.1.2. Izejas informācijas apraksts 8](#_Toc74045217)

[2.1.3. Ārējās informācijas apraksts 9](#_Toc74045218)

[2.2. Funkcionālās prasības 11](#_Toc74045219)

[2.3. Nefunkcionālās prasības 12](#_Toc74045220)

[3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU PAMATOJUMS 13](#_Toc74045221)

[4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA 15](#_Toc74045222)

[4.1. Sistēmas struktūras modelis 15](#_Toc74045223)

[4.1.1. Sistēmas arhitektūra 15](#_Toc74045224)

[4.1.2. Sistēmas ER modelis 17](#_Toc74045225)

[4.2. Funkcionālais sistēmas modelis 20](#_Toc74045226)

[4.2.1. Datu plūsmu modelis 20](#_Toc74045227)

[5. DATU STRUKTŪRAS APRAKSTS 41](#_Toc74045228)

[6. LIETOTĀJA CEĻVEDIS 47](#_Toc74045229)

[6.1. Sistēmas prasības aparatūrai un programmatūrai 47](#_Toc74045230)

[6.2. Sistēmas instalācija un palaišana 49](#_Toc74045231)

[6.3. Programmas apraksts 51](#_Toc74045232)

[NOBEIGUMS 64](#_Toc74045233)

[INFORMĀCIJAS AVOTI 65](#_Toc74045234)

[1. Ekrānformu piemēri 67](#_Toc74045235)

[2. pielikums ER diagramma 72](#_Toc74045236)

[3. pielikums tabulu saišu shēma 73](#_Toc74045237)

# IEVADS

Jebkuru darbību var izpildīt ātrāk to automatizējot. Šis princips attiecas, ne tikai, uz ražošanu un fizisku produktu izstrādi, bet arī uz pašu cilvēku darba gaitu, piemēram, mācīšanos. Lielu daļu laika, skolotāji pavada labojot darbus un tikai tad, kad darbs ir ticis izlabots var sākt analizēt, kas tieši nav bijis pareizi - tas ir laikietilpīgs darbs, kuru varētu pavadīt lietderīgāk, lai nodarbības būtu kvalitatīvākas. Apgūstot tādu priekšmetu, kā programmēšana, darbu labošana kļūst vēl sarežģītāka, jo jāņem vērā ir, ne tikai, atbilžu rezultāti, bet arī algoritma efektivitāte, laika un atmiņas patēriņš, nemaz nerunājot par to, ka, lai pilnībā pārbaudītu sarežģītus algoritmus ir nepieciešams veikt ļoti daudz testus, kas labākajā gadījumā ir ļoti ilgs darbs un sliktākajā gandrīz neiespējams vienam cilvēkam, pie tam, nav garantijas, labotājs nav palaidis garām kādu, tā saucamo, “edge-case” un izsecinājis uzdevumu, kā korekti izpildītu, lai gan patiesībā atrisinājums nav pareizs.

Šādu problēmu atrisināšanai izmanto “automatizētus testa serverus”, kas kompilē lietotāja iesūtīto kodu un salīdzina to, ar jau, iepriekš izveidotiem, testa piemēriem un beigās uzrāda vai iesūtījums ir iekļāvies kritērijos, vai arī nav.

Kvalifikācijas darba ietvaros izstrādāju, šādu sistēmu priekš SIA “Pirmā programmēšanas skola”, lai gan ir līdzīgi, bezmaksas, publiski pieejami produkti, piemēram Clevercode.lv un replit.com, uzņēmums saskārās ar limitācijām, ko var atrisināti, tikai, personalizēta sistēma, piemēram clevercode neatbalsta uzdevumu iesūtīšanu Python valodā, kas ir pašlaik visdraudzīgākā valoda iesācējiem. Par cik lielākā daļa nodarbību ir vērstas uz algoritmizāciju un sporta programmēšanu, ir svarīgi pievienot savus uzdevumus, ar saviem testa piemēriem un kritērijiem, bet sistēmas, kas tika izmantotas līdz šim atļauj pildīt uzdevumus tikai no dotā uzdevumu krājuma. Rezultātā, tika izlemts, ka vislabāk būtu izveidot personalizētu sistēmu tieši priekš SIA “Pirmā programmēšanas skola”

# UZDEVUMA NOSTĀDNE

Kavalifikācijas darba uzdevums ir izveidot mācību sistēmu ar automatizētu testa serveri, priekš SIA “Pirmā programmēšnas skola”. Ar tās palīdzību skolēni var iesūtīt savus uzdevumu risinājumus un sistēma atrisina, vai dotais algoritms iekļaujas uzdevuma nosacījumos. Lietotājam ir iespēja ne tikai iesūtīt uzdevumu, bet arī apskatīt rezultātu, pareizi atrisinātos uzdevumus, neatrisinātos uzdevumus un savu kopējo statistiku. Admin lietotāja galvenais uzdevums ir pievienot un pārvaldīt uzdevumus, kā arī lietotājus - piešķirt lietotājam admin statusu vai dzēst lietotāju.

SIA “Pirmā programmēšanas skola” mācību sistēmai ir jāizpilda vairākas funkcionalitātes:

Uzdevuma datu (Nosaukums, kods, definīcija, ievaddati, izvaddati, kritēriji) un tā testa piemēru izveidošana, dzēšana, labošana, ko var darīt tikai admin lietotājs.

Katrs lietotājs var iesūtīt uzdevumus - iesūtījums tiek kompilēts un salīdzināts ar testa piemēriem, tādējādi secinot vai uzdevums ir bijis pareizs.

Iesūtījumu un atrisinājumu uzglabāšana - lietotājs var apskatīt savus iesūtījumus.

Pieslēģšanās un reģistrēšanās sistēma, kas ļauj datu apstrādi balstoties uz lietotāja lomas - admins vai lietotājs,

Lietotāju reitinga aprēķināšana - lieotājs var salīdzināt sevi ar pārējiem lietotājiem.

Lietotāja profila informācijas labošana.

Lietotāju pārvaldīšana, ko var darīt tikai admin lietotājs - mainīt lietotāja privilēģijas vai dzest lietotāju.

# 2. PRASĪBU SPECIFIKĀCIJA

## 2.1. Ieejas un izejas informācijas apraksts

2.1.1. Ieejas informācijas apraksts

Sistēmā tiks nodrošināta šāda ieejas informācijas apstrāde

1. Informācija par **lietotājiem** sastāvēs no sekojošiem datiem.
   * Vārds – burtu ieraksts ar garumu līdz 32 simboliem
   * Lietotājvārds – burtu ieraksts ar izmēru līdz 32 simboliem
   * Epasta adrese – burtu ieraksts, kam jābūt epasta formātā ar garumu, ar vismaz 8 simboliem, līdz 64 simboliem
   * Parole – burtu ieraksts ar izmēru līdz 64 simboliem
   * Admins – būla tipa mainīgais, kas norāda uz admina statusu – šo vērtību var mainīt tikai admin panelī
   * Skolotajs – būla tipa mainīgais, kas norāda uz skolotāja statusu – šo vērtību var mainīt tikai admin panelī
2. Informācija par **jaunumu ierakstiem** sastāvēs no sekojošiem datiem
   * Ieraksta nosaukums – burtu ieraksts ar garumu līdz 128 simboliem
   * Ieraksta apraksts – burtu ieraksts bez limita, kas tiek ievadīts caur TinyMCE teksta redaktoru un tiek saglabāts kā HTML kods ar teksta dizaina iespējotiem parametriem
3. Informācija par **uzdevumiem** sastāvēs no sekojošiem datiem
   * “Pievienot ar polygon” – zip faila pievienošana, kas aizpilda uzdevuma datu laukus, ja failā tiek atrasti faili, kas izveidoti ar uzdevumu izstrādes, polygon, sistēmas palīdzību
   * Uzdevuma kods – burtu ieraksts ar garumu līdz 64 simboliem
   * Uzdevuma nosaukums – burtu ieraksts ar garumu līdz 128 simboliem
   * Ievaddatu rakturojums – burtu ieraksts bez garuma limita
   * Izvaddatu raksturojums – burtu ieraksts bez garuma limita
   * Uzdevuma definīcija – burtu ieraksts bez garuma limita
   * Laika limits (Sekundēs) – skaitlis ar komatu
   * Atmiņas limits (Megabaitos) – skaitlis ar komatu
   * Punkti – skaitlis bez komata
   * Grūtība – skaitlis bez komata
   * Tagi – Atkarībā no mājaslapas sistemā saglabātajiem tagiem var izvēlēties vienu vai vairākus tagus ko piesaistīt uzdevumam caur izvēlnes elementu
   * Paraugdati – bezgalīgi daudz pievienojami, divu bezlimita burtu ierakstu pāri, kas attiecīgi raksturo testa ievaddatus un vēlamos izvaddatus
4. Informācija par **uzdevumu meklēšanu** sastāvēs no sekojošiem datiem
   * Uzdevuma nosaukums – bezlimita simbolu virkne
   * Filtrs – izvēles logs, kurā var izvēlēties pēc kā kārtot uzdevumus
   * Secība – izvēles logs, kurā var izvēlēties kādā secībā uzdevumus atspoguļot – augošā, vai dilstošā
5. Informācija par **Tagiem** sastāvēs no sekojošiem datiem
   * Nosaukums – burtu ieraksts ar garumu līdz 64 simboliem
   * Apraksts – burtu ieraksts bez garuma limita
   * Krāsa – ar krāsu selektoru izvēlama krāsa, kas saglabājas kā izvēlētās krāsas heksadecimālais kods
6. Informācija par **Iesūtījumiem** sastāvēs no sekojošiem datiem
   * Valoda – izvēles logs kurā iespējams izvēlēties, kādā valodā iesūtīt uzdevumu
   * Kods – bezlimita burtu ieraksts
7. Informācija par **Banera bildēm** sastāvēs no sekojosiem datiem
   * Banera bilde – bildes fails

2.1.2. Izejas informācijas apraksts

1. Uzdevuma iesūtījuma rezultāts. Iespējams apskatīt pārbaudīta iesūtījuma rezultātus:
   * Vai uzdevums ir pareizi atrisināts
   * Iegūto punktu skaitu attiecībā pret kopējo punktu skaitu
   * Katram pārbaudītajm testa piemēram atsevišķi:
     + Vai tests izpildīts pareizi – norādīts ar krāsu ap testu
     + Testa kārtas numuru
     + Laiku (sekundēs) kādā izpildīts tests
     + Atmiņu (megabaitos) ar kādu izpildīts tests
     + Ko dotais kods izvadīja dotajā testā (ievaddati netiek norādīti)
2. Profils – lietotāja statistika
   * Tiek uzrādīti visi lietotāja neatrisinātie uzdevumi (Uzdevumi, kuriem veikts iesūtījums, bet nav bijuši atrisināti)
   * Lietotāja atrisinātie uzdevumi
   * Lietotāja iegūtie punkti
   * Lietotāja uzdevumu reitings (Cik procentuāli no mēģinātiem uzdevumiem lietotājs uzdevumus ir atrisinājis)
   * Lietotaja statistika kā grafiks, kurā ar atrisinājumu laikiem norādīts, cik atrisinājumi ir iegūti
3. Jaunumu ieraksti – jaunumu ieraksti tiek uzrādīti, kā teksts ar norādīto dizainu, kāds tika izveidots teksta redaktorā, kad ieraksts tika pievienots, kā arī, izveidošanas laiks, ieraksta nosaukums un lietotāja lietotājvārds, kas ierakstu izveidoja
4. Banera bildes – banera bildes, jau saglabātas serverī, tiek uzrādītas sākumlapas banerī
5. Reitings – lietotāji tiek sakārtoti pēc iegūtajiem punktiem punktiem un tiek uzrādīts
   * Vieta kopējā lietotāju reitingā
   * Litotāja lietotājvārds
   * Atrisinātie uzdevumi
   * Neatrisinātie uzdevumi
   * Atrisinājumu reitings
   * Punktu skaits
6. Uzdevumu meklētājā uzdevumi tiek sakārtoti skatoties uz parametriem, ko lietotājs ir noteicis – pēc kā filtrēt (Pievienošanas laiks, grūtība, atrisinājumu reitings), secība (augoša/dilstoša), mekētaja ieraksta (Uzdevumi meklētājā tiek meklēti, pēc uzdevuma nosaukuma)

2.1.3. Ārējās informācijas apraksts

Sistēma izmanto Judge0 sistēmu caur RapidAPI hostu. Judge0 sistēma nodrošina uzdevuma koda kompilēšanu. Ar HTTP request palīdzību uz Judge0 tiek nosūtīti sekojoši dati:

* Valodas kods kurā iesūtījums ir jāizskata
* Iesūtījuma kods
* Masīvs ar testu ievaddatiem, kas jāievada iesūtītajam kodam

Hosts atgriež unikālu “token” un algoritms serverī atkārtoti iegūst iesūtījuma rezultātu, līdz rezultāts ir sakompilēts (Kamēr uzdevums nav ticis sakompilēts Hosts atgriež ziņu “in queue” vai “processing”) Kad uzdevums ir sakompilēts tiek iegūts masīvs, kur katram testam ir iegūtas sekojošās vērtības:

* Izvaddati, kas radušies dotajam kodam ievadot dotā testa ievaddatus
* Laiku (Sekundēs), cik ilgi tas aizņēma
* Atmiņu (Kilobaitos), cik daudz tas ir aizņēmis

Tad jau lietotāja serverī tiek veikta salīdzināšana ar gaidāmajiem izvaddatiem un loģika, kas nosaka vai iesūtījums ir bijis pareizs.

## 2.2. Funkcionālās prasības

1. Sistēmai jebkuram mājaslapas apmeklētājam, pat ja tas nav pieslēdzies, ir jānodrošina sekojošās funkcionalitātes
   1. Apskatīt sākumlapu – banera bildes un jaunumu ierakstus
   2. Apskatīt jebkuru uzdevumu
   3. Apskatīt reģistrēto lietotāju reitingu
   4. Apskatīt jebkura lietotāja profila statistiku
2. Jānodrošina lietotāja reģistrāciju.
   1. Ja ievadlauku prasības ir ievērotas un visi lauki aizpildīti, datbubāzē jāsaglabā lietotājs bez admina statusa.
3. Jānodrošina lietotāja autorizāciju.
   1. Ja lietotājs ir reģistrēts, sistēmai ir jāpieslēdz lietotājs ievadot lietotāju un paroli
4. Ja lietotājam ir admina statuss sistēmai ir jāpiedāvā piekļuve admina iespējām
   * 1. Izveidot uzdevumu – izveidot uzdevumu var tikai tad, ja visi lauki ir bijuši pareizi aizpildīti
     2. Labot uzdevumu – labot uzdevumu var tikai tad, ja visi lauki ir bijuši pareizi aizpildīti
     3. Dzēst uzdevumu
     4. Izveidot jaunumu ierakstu ja visi lauki ir aizpildīti
     5. Dzēst jaunumu ierakstu
     6. Piekļūt administratora panelim
        1. Rediģēt lietotājus
        2. Rediģēt tagus, ja visi lauki ir aizpildīti
        3. Rediģēt banera bildes – banera bildei jābūt kādā no bildes faila formātiem
5. Jānodrošina uzdevumu iesūtīšanu
   1. Ja lietotājs ir pieslēdzies sistemai, tai ir jāļauj iesūtīt jebkurš uzdevums, ja koda lauks nav bijis tukšs
6. Profila datu apstrāde – lietotājam bez admina statusa ir jābūt piekļuvie sekojošajiem datiem
   1. Ja lietotājs ir pieslēdzies sistēmai, sistēmai ir jāļauj apskatīt savus profila statistiku
   2. Ja lietotājs ir pieslēdzies sistēmai, jāļauj labot profila datus, ja datu lauki aizpildīti pareizi
   3. Lietotājam pie katra uzdevumu jābūt iespējai apskatīt visus savus iesūtījumus
   4. Lietotājam pie katra uzdevuma jābūt iespējai apskatīt visus savus veiksmīgos iesūtījumus, jeb atrisinājumus

## 2.3. Nefunkcionālās prasības

1. Kreisajā pusē jāatrodas aizveramai izvēlnei
2. Augšējā labajā stūrī jāatrodas lietotājvārdam, uz kura uzspiežot jāatveras izvēlnei, caur kuru var nonākt lietotāja profilā, vai, ja ir tiesības, admin panelī
3. Visai lietotāja saskarsnei jābūt latviešu valodā
4. Galvenajām sistēmas dizaina krāsām jābūt, SIA “Pirmā Programmēšānas Skola” stilā – balts, gaiši zils, melns, tumši pelēks.
5. Kreisajā pusē, lietotāja izvēlnē, augšējā stūrī jāatrodas PPS logotipam
6. Banera bildēm, galvenajā lapā ir jāietilpst banerī – tās nedrīkst plūst pāri, vai deformēt dizainu
7. Datubāzes datu apstrādes pogām jābūt labi atšķiramām – sarkana, lai dzēstu, zaļa, lai pievienotu, dzeltena, lai labotu

# 3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU PAMATOJUMS

1. Vides
   1. PhPstorm – Pati programma izstrādāt Jetbrains PHPStorm vidē, jo videi ir ļoti ērta un vienkārša lietotāja saskarsne, tiek uzrādīti faili, kas ir un nav pievienoti git datuplūsmai un šī vide atbalsta sintaksi visām nepieciešajām valodām, kas tika izmantotas projektā.
   2. Windows Cmd – Liela daļa programmas tika testēta un pārbaudīta caur windows termināli, jo izmantotajam ietvaram ir pievienots iebūvēts testa serveris, kurā var modificēt datubāzes caur termināļa komandām, kā arī, lai palaistu speciālas ietvara komandas.
2. Ietvari
   1. Laravel – ietvars, kas tika izmantots mājaslapas veidošanā. Ietvarā ir ļoti ērta modeļu sistēma un darbs ar datubāzēm, datu atlasi un pāradresēšanu starp mājaslapas daļām ir ļoti ērts. Laravel ir paredzēts web aplikāciju izveidei un atbild par izmantoto valodu savstarpēju darbību.
   2. VUE.js – JavaScript ietvars, kas ļauj izveidot komponentus – atsevišķus moduļus, kas var tikt atkārtoti ievietoti jebkurā mājaslapas daļā, kā arī, par cik, šī ir javascript valoda, šis ietvars ļauj veidot dinamiskus elementus, kuriem nav nepieciešams pārlādēt mājaslapu, mainītu kāda elementu kopuma, piemēram ievadlauku, skaitu.
   3. AJAX – JavaScript metožu kopums, kas tiek izmantots, lai sūtītu komandas uz datubāzi bez mājaslapas pārlādēšanas, tas ir ērti izmantojams VUE.js komponentos, kur AJAX darbības, var rakstīt tieši iekšā komponenta funkcijās.
   4. Bootstrap – CSS ietvars ar sistēmu, kam ir iestrādātas css klases, ar kurām ērti pārveidot mājaslapas dizainu.
3. Valodas
   1. HTML – galvenā valoda, kas izmantota mājaslapas strukturizēšanai.
   2. CSS – galvenā valoda, kas izmantota mājaslapas dizaina noformēšanai.
   3. JS – valoda izmantota VUE.js ietvarā, ar šo tiek izveidoti dinamiskie komponenti.
   4. PHP – valoda, kas izmantota servera daļā – laravel kontolieros un modeļos.
   5. Eloquent – laravel relāciju-objektu kartētājs. Tiek izmantots, lai ērti, ar laravel modeļu sistēmu, piekļūtu un mainītu datubāzes datus.
   6. Blade – blade tika rakstīts galvenais front-end daļas kods. Blade ir “templating engine”, tā sastāv no html koda, lai strukturizētu mājaslapu, bet tajā var ievietot arī PHP kodu, ar kuru var apstrādāt lapas datus, vai pat iegūt jaunus datus neizmantojot kontroliera starpniecību.
4. Plugini
   1. CodeMirror – JavaScript koda redaktors. CodeMirror nav koda kompilācijas funkcionalitātes – tas tiek izmantots tikai koda noformēšanai. CodeMirror ir pieejams uzstādīt vairāku programmēšanas valodu sintakses, lai tas iekrāsotu koda atslēgvārdus.
   2. TinyMCE – JavaScript teksta redaktors. Iespējams ar javascript pārveidot textarea redaktoru, par TinyMCE teksta redaktoru, ar kuru ir iespējams noformēt tekstu, piemēram, mainīt burtu krāsu, lielumu, uzlikt bold un italic efektus. Teksta vērtība, ko iegūst no ievadlauka ir html kods ar noformētā teksta parametriem.
   3. MorrisJS – Grafiku veidošanas funkcijas, ar kurām JavaScript valodā var veidot dažādus, vizuālus grafikus.
   4. AdminLTE V2.0 – Programmā tiek izmantota AdminLTE veidne (“Template), kura nostāda mājaslapas galveno daļu izvēlņu izkārtojumu – aizverama navigācijas izvēlne kreisajā pusē un lietotāja izvēlne augšējā, kreisajā stūrī.

# 4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

## 4.1. Sistēmas struktūras modelis

4.1.1. Sistēmas arhitektūra

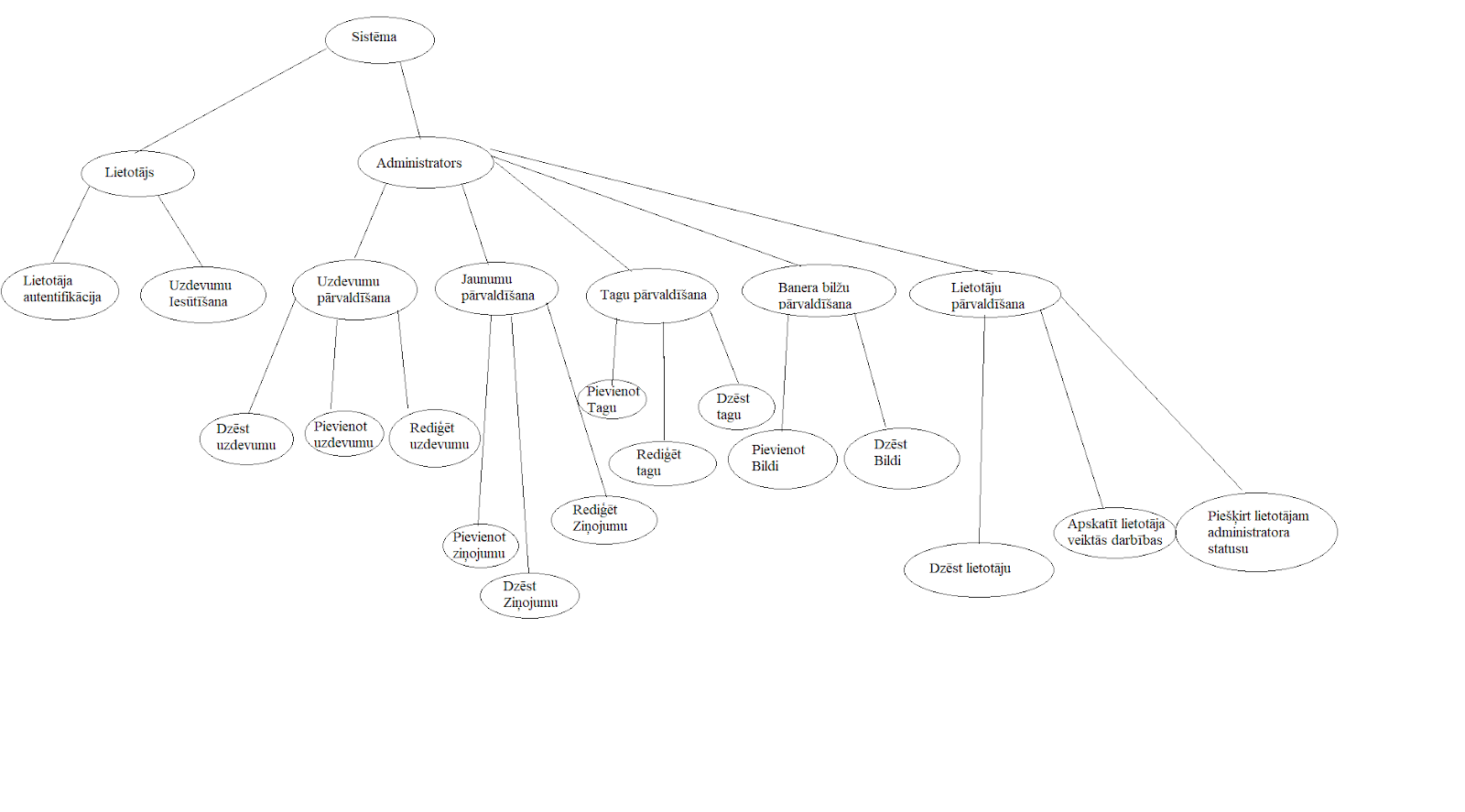
Kopumā PPS sistēma sastāvēs no 3 daļām.

1. Administratora daļa.
2. Lietotāja daļa.

**Administratora daļa** ir paredzēta, lai varētu rediģēt lapas galveno saturu. Administratora loma ir paredzēta, lai mainītu informāciju, kas redzama lapā un, kas atrdoas datubāzē - piem. pievienot vai dzēst uzdevumus. Administratora galvenais uzdevums ir pārvaldīt lapas saturu - pievienot vai dzēst jaunumu ziņojumus, izveidot jaunus uzdevumus un pārvaldīt lietotājus, gadījumā, ja tiek pamanīti mēģinājumi, uzdevumu izpildē šmaukties.

**Lietotāja daļa** ir paredzēta, lai būtu iespējams izpildīt lapas paredzētās pamatfunkcijas - pildīt uzdevumus. Lietotājam ir pieejams apskatīt jaunumu lapu, piekļūt visiem publiski redzamajiem uzdevumiem, veikt iesūtījumu, kādā no piedāvātajām valodām, un apskatīt rezultātu, kā arī redzēt sevi kopējā reitingā.

Sistēmas arhitektūras diagrammu var apskatīt 1. Pielikumā



4.1..att. Sistēmas arhitektūras diagramma

4.1.2. Sistēmas ER modelis

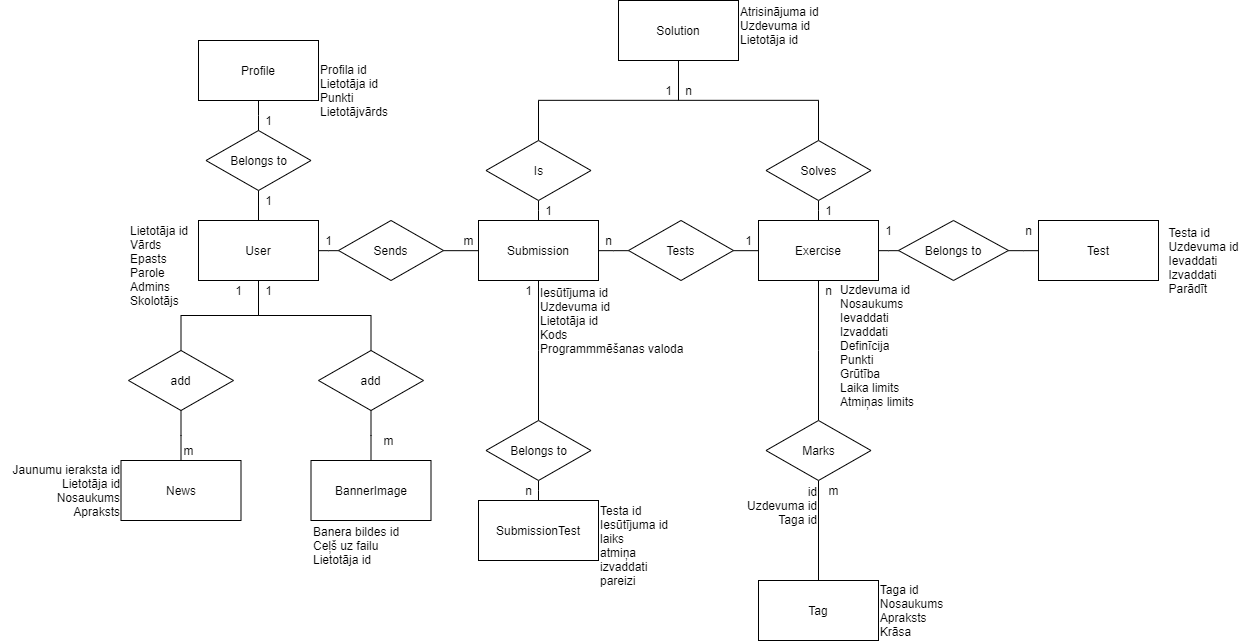
Datu bāzes projektēšanā datu kopu un saišu starp tām attēlošanai tika lietota realitāšusaišu diagramma, kas sastāv no divu veidu objektiem – entītēm (loģiskais objekts) un relācijām (attiecības).

Datu bāzes ER modelis , kas ir dots 1.pielikumā, sastāv no sekojošām entītijām:

* **“Exercise” -** apraksta uzdevumus sistēmā. Tās atribūtu kopums sevī ietver uzdevuma nosaukumu, kodu, definīciju, punktus, laika limitu un atmiņas limitu.
* **“Submission”** - apraksta uzdevuma iesūtījumus. Tās atribūtu kopums sevī ietver iesūtīto kodu un izvēlēto programmēšanas valodu.
* **“Tag” -** apraksta uzdevumu identifikatorus. Tās atribūtu kopums sevī ietver nosaukumu, aprakstu un identifikatora krāsu.
* **“Test”** - apraksta uzdevumu testus. Tās atribūtu kopums sevī ietver testa ievaddatus, izvaddatus un boleānu, kas pārbauda, vai tests būs publiski redzams.
* **“Solution”**  - apraksta pareizi izpildītos iesūtījumus. Tās atribūtu kopums sevī ietver norādes (Foreign key) uz korekti izpildītajiem uzdevumiem.
* **“SubmissionTest”** - apraksta iesūtījuma veiktos testus. Tās atribūtu kopums sevī ietver izpildes laiku un atmiņu, izvaddatus un boleānu, kas pārbauda vai tests izpildīts pareizi.
* **“User”** - apraksta privāto informāciju par lietotāju. Tās atribūtu kopums sevī ietver lietotāja vārdu, epastu, paroli un boleānu, kas atbild par to vai lietotājs ir admins.
* **“Profile”** - apraksta publiski pieejamo informāciju par lietotāju. Tās atribūtu kopums sevī ietver lietotāja kopējos punktus un lietotājvārdu.
* **“News”** - apraksta ziņu informāciju. Tās atribūtu kopums sevī ietver nosaukumu un aprakstu.
* **“BannerImage”** apraksta banerī pievienotās bildes. Tās atribūtu kopums sevī ietver bildes atrašanās vietas linku.

Datu bāzes relācijas parada kā savstarpēji ir savienotas divas vai vairākas entītijas:

* Starp User un Profile ir relācija 1 pret 1, jo vienam lietotājam var būt tikai viens profils
* Starp User un News ir relācija 1 pret daudziem, jo viens lietotājs var pievienot vairākas ziņas, bet ziņas autors var būt tikai viens
* Starp User un BannerImage ir relācija 1 pret daudziem, jo viens lietotājs var pievienot vairākas Bildes, bet bildi var pievienot tikai viens cilvēks
* Starp User un Submission ir saite viens pret daudziem, jo viens lietotājs var veitk daudz iesūtījumus, bet iesūtījums var piederēt tikai vienam lietotājam.
* Starp SubmissionTest un Submission ir saite viens pret daudziem, jo vienam iesūtītjumam var būt vairāki testi, bet viens tests var testēt tikai vienu iesūtījumu
* Starp Solution un Submission ir saite 1 pret 1, jo vienam iesūtījumam var būt tikai viens atrisinājums
* Starp Exercise un Submission ir saite viens pret daudziem, jo uzdevumam var būt vairāk iesūtījumi, bet iesūtījumu var iesūtīt tikai vienam uzdevumam.
* Starp Exercise un Solution ir saite viens pret daudziem, jo vienam uzdevumam var būt vairāki atrisinājumi, bet viens atrisinājums var atrisināt vienu uzdevumu.
* Starp Exercise un Tag ir saite daudzi pret daudziem, jo vienam uzdevumam var būt vairāki identifikatori un viens identifikators var identificēt vairākus uzdevumus
* Starp Exercise un Test ur saite viens pret daudziem, jo vienam uzdevumam var būt vairāki testi, bet tests var piederēt vienam uzdevumam.



4.2.. att Sistēmas ER modeļa shēma

## 4.2. Funkcionālais sistēmas modelis

4.2.1. Datu plūsmu modelis

**1. Lietotāja reģistrācija**

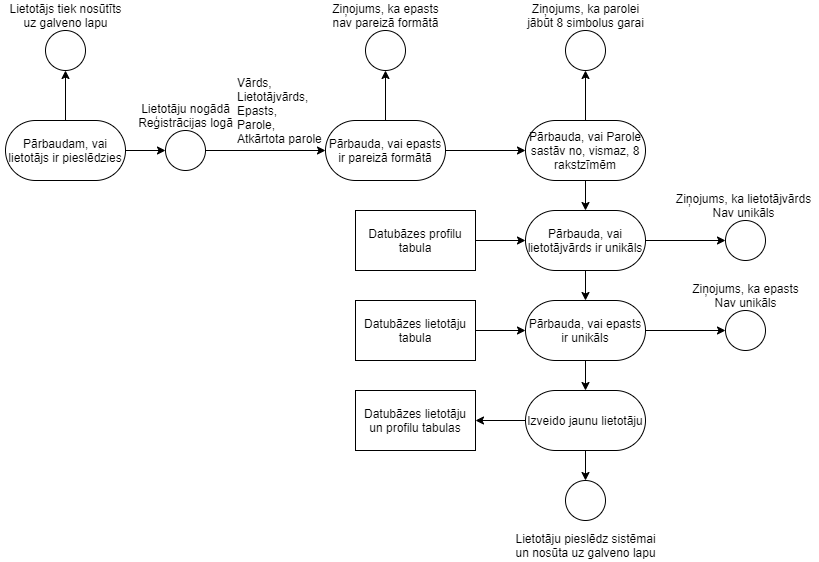
Lietotāja reģistrācijas pamatuzdevums ir izveidot sistēmā jaunu lietotāju, bez jebkādām privilēģijām, bet nodrošināt visas lietotāja lomas pamatfunkcionalitātes.

Laravel autentifikācijas sistēma pārbauda, vai lietotājs jau ir pieslēdzies sistēmā, ja lietotājs ir, tad pieslēgšanās un reģistrācijas pogas tiek slēptas un tiek uzrādīta iespēja atslēgties no sistēmas. Ja lietotājs kautkādā veidā, tomēr mēģina nokļūt pieslēgšanās, vai reģistrācijas ekrānā, piemēram ievadot internetpārlūkprogrammas meklētājā konkrētu linku, lietotājs tiek atgriezts uz uzdevumu saraksta lapu, kas kalpo kā mājaslapas galvenā daļa.

Ja lietotājs nav pieslēdzies, tas tiek nogādāts reģistrācijas logā. Reģistrācija sastāv no 5 laukiem – vārda, lietotājvārda, epasta adreses, paroles un atkārtotas paroles (apstiprināt paroli lauks). Ja visi lauki ir tikuši aizpildīti, tiek pārbaudīts vai tie ir aizpildīti pareizā formātā – epasta laukam ir jābūt kā epastam – ar atdalošu “@” simbolu un punktu, kas atdalītu epasta domēna sufiksu. Parolei ir jāsastāv no vismaz astoņām rakstzīmēm. Ja parasības netiek ievērotas, pie laukiem uzrādās kļūdu ziņojumu, kas norāda uz nekorekti aizpildītajiem laukiem.

Ja visas prasības ir ievērotas, ievadītie dati tiek salīdzināti ar datubāzes ierakstiem, kur tiek pārbaudīts, vai lietotājvārds un epasts ir unikāls, ja nē tiek uzrādīti kļūdu ziņojumi, kas norāda uz sakritību un, lai lietotājs izvēlas citu paroli un lietotājvārdu.

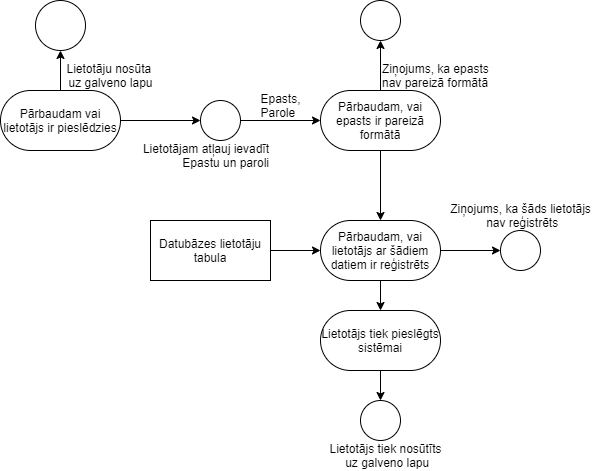
Ja visi šie kritēriji ir ievēroti datubāzē tiek izveidots jauns lietotājs, bez īpašām privilēģijām.



4.3.att. Lietotāja reģistrācijas datu plūsmu diagramma

**2. Lietotāja pieslēgšanās**

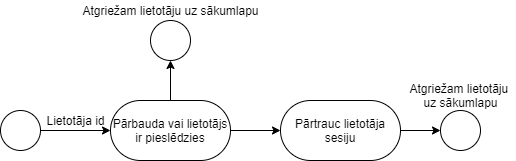
Kad lietotājs ir izveidojis profilu, viņam ir jāļauj pieslēgties sistēmai. Ja lietotājs nav šobrīd pierakstījies sistēmā, ir iespējams nonākt pieslēgšanās logā, kur tiek pieprasīts epasts un parole. Ja epasts ievadīts korektā formātā, tiek pārbaudīts vai lietotājs ar šādu epastu un paroli eksistē datubāzē. Ja lietotājs tiek atrasts, viņš tiek pieslēgts sistēmai, ja nē, tad tiek uzrādīts, ka dati nav ievadīti pareizi. Ja lietotājam ir admin privilēģijas, lapā tiek uzrādītas pogas ar, tikai adminiem, pieejamām darbībām.



4.5.att. Lietotāja pieslēgšanās datu plūsmu diagramma

**3. Atslēgšanās no sistēmas**

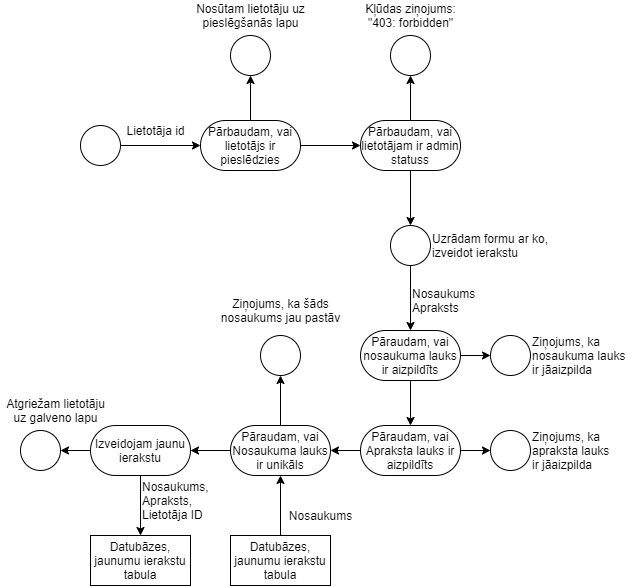
Ja lietotājs ir pieslēdzies sistēmai, tad augšējā, kreisajā stūrī nospiežot uz sava lietotājvārda atveras izvēlna, kurā atrodas izrakstīšanās poga, uz šīs pogas uzspiežot lietotājs, tiek atslēgts.



4.6 att. Lietotāja atslēgšanas datu plūsmu diagramma

**4. Jaunumu ierakstu izveidošana**

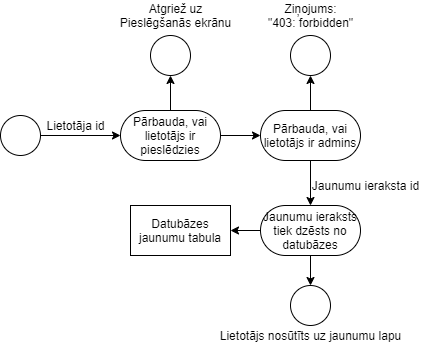
Ja lietotājam ir piešķirts admina statuss un lietotājs pašlaik ir pieslēdzies sistēmā, ienākot lapā “jaunumi” ar “blade directive” palīdzību, tiek pārbaudīts vai lietotājm ir admina statuss, ja ir, zem banera bildes jāuzrādās pogai “izveidot jaunu ierakstu” (Lietotājiem, bez admina tiesībām, šāda poga nerādās), nospiežot uz šīs pogas lietotājs tiek nosūtīts uz lapu, kur tiek uzrādīta forma, kas sastāv no diviem laukiem – nosaukums un apraksts. Tiek pārbaudīts, vai abi lauki ir azipildīti un vai ieraksta nosaukums nesakrīt ne ar vienu citu datubāzē, ja nē datubāzē tiek izveidots jauns “news” objekts, kas pēc tam tiek uzrādīts jaunumu lapā. Ja kritēriji netiek ievēroti, pie laukiem tiek uzrādīti paziņojumi ar to, kas ir jāizlabo, lai tiktu atļauts ierakstu izveidot.



4.7. Jaunumu ieraksta izveidošanas datu plūsmas diagramma

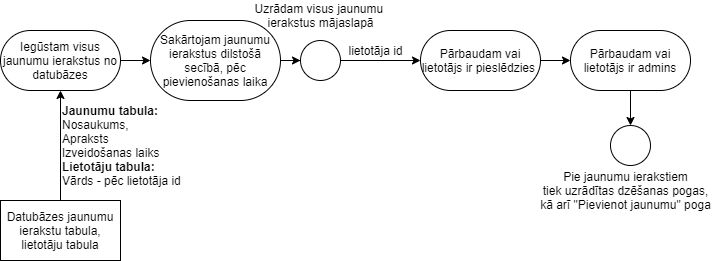
**5. Jaunumu ierakstu dzēšana**

Ja lietotājs ir pieslēdzies sistēmai un tam ir admin statuss, tad zem katra jaunumu ieraksta, tiek uzrādīta “dzēst ierakstu” poga. Nospiežot uz šīs pogas uz datubāzi, tiek nosūtīts pieprasījums, kas dzēš ierakstu kura ID, tika norādīts pie dzēšanas pogas.



4.8. att. Jaunumu ieraksta dzēšanas datu plūsma

**6. Jaunumu ierakstu attēlošana**

“Jaunumu lapa” sastāv no divām galvenām daļām – jaunumu ierakstiem un banera bildēm. Ieslēdzot lapu no datubāzes tiek iegūti visi jaunumu ieraksti, sakārtojot no jaunākā uz vecāko un tiek uzrādīti, kā atsevišķi logi janumu lapā.

4.9. att Jaunumu ierakstu attēlošanas datu plūsma

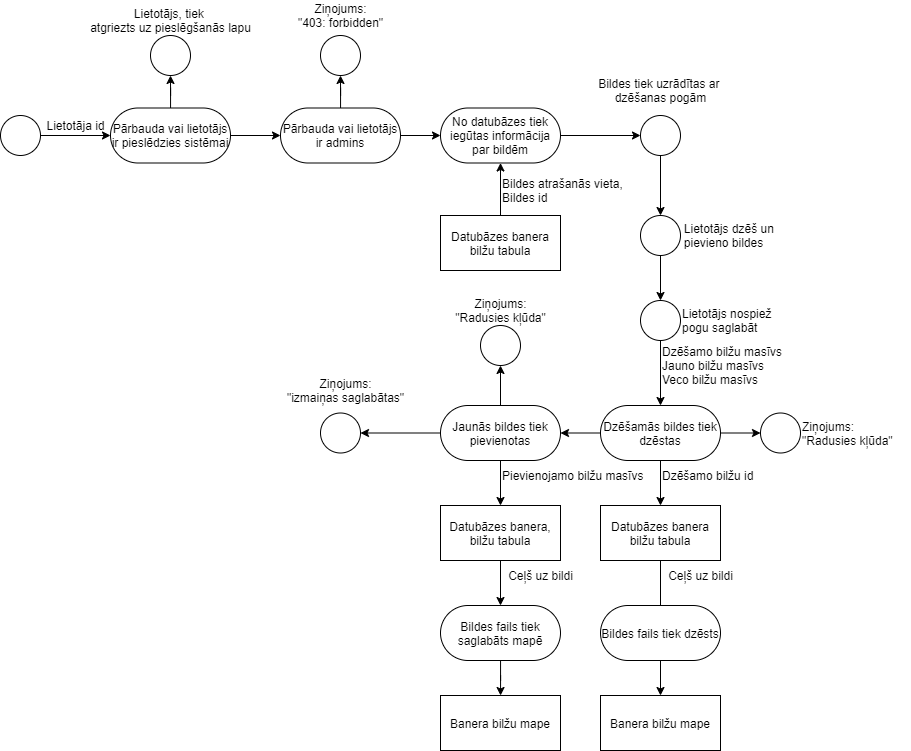
**7. Banera bilžu rediģēšana**

Ja lietotājs ir pieslēdzies sistemai un tam ir admin statuss, tieši virs galvenā bilžu banera jāatrodas pogai “rediģēt banera bildes” uz kuras nospiežot lietotajs nonāk, banera bilžu redaktorā, kur ir iespējams pievienot jaunu bildes failu. Failu meklētājā, tiek uzrādīti tikai faili, kuru formātu var kategorizēt kā bildi, tāpēc citus nav iespējams pievienot.

Ienākot banera bilžu rediģēšanas lapā, no datubāzes tiek iegūtas visas esošo banera bilžu adreses un bildes tiek uzrādītas. Lietotājs var pievienot bezgalīgi daudz bildes un noņemt no saraksta jau esošās, bet izmaiņas tiek saglabātas datubāzē tikai tad, kad tiek nospiesta poga “saglabāt”. Visas informācija par bildēm tiek nosūtīta uz kontrolieri, kur tiek sadalītas:

1. Vecās bildes, kuras jādzēš
2. Jaunās bildes kuras jāpievieno
3. Vecās bildes, kuras nav jāmaina

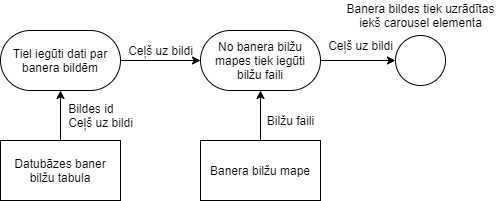
Katram bildes objektam, tiek dots identifikators, kas atšķir, kas ar šo bildi jādara. Ar katru bildi, atkarībā no tās statusa tiek veikta nepieciešamā operācija – ja jādzēš, tad bilde tiek dzēsta no banera bilžu mapītes, un no datubāzes ir jādzēš ieraksts ar bildes atrašanās vietu, ja bilde jāpievieno, tad bilde tiek saglabāta banera bilžu mapītē, ja bilde jau atradās datubāzē un netiek dzēsta, tad ar to nekas nav jādara.



4.10. att. Banera bilžu rediģēšanas datuplūsma

**8. Banera bilžu attēlošana**

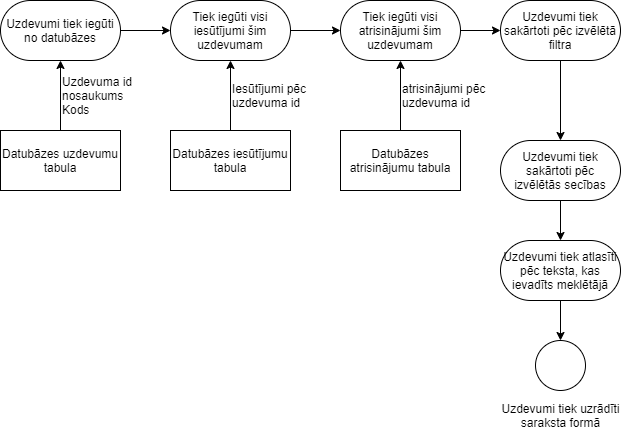
“Jaunumu lapa” sastāv no divām galvenām daļām – jaunumu ierakstiem un banera bildēm. Ieslēdzot “jaunumu lapu” no datubāzes tiek iegūti visi ieraksti ar informāciju par banera bildēm – banera bildes atrašanās vietu un tās kārtas numuru. Balstoties uz Banera bildes atrašanās vietu “Bootstrap carousel” elementā tiek attēlotas banera bildes.



4.11. att Banera bilžu attēlošanas datuplūsma

**9. Uzdevumu attēlošana**

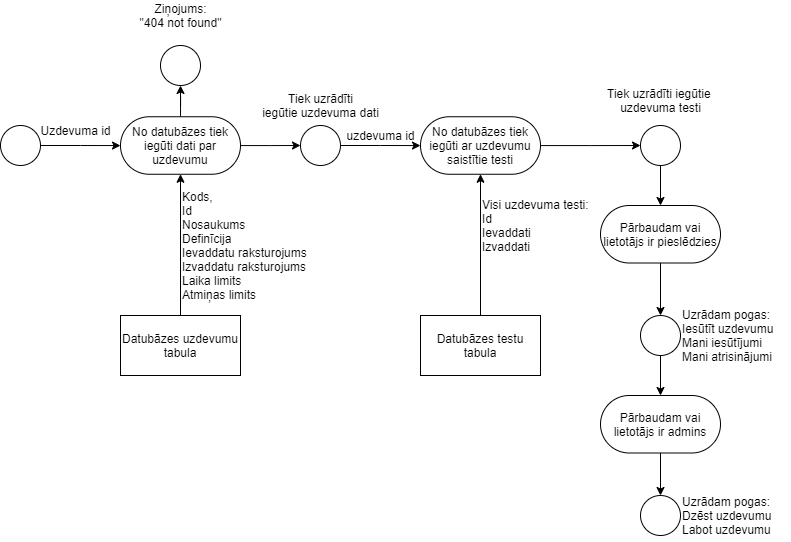
Galvenais sistēmas uzdevums ir lietotājiem ļaut pildīt uzdevumus. Atverot uzdevumu lapu no datubāzes tiek izgūti visi uzdevumi. Uzdevumu dati tiek iedoti VUE.js komponentam, kuram ir pievienots meklētājs un filtrs, ko lietotājs var dinamiski mainīt. Tiek apskatīta filtra izvēlētā vērtība, pēc kuras tiek sakārtoti dati, kā arī pēc izvēles, vai datus kārtot augošā, vai dilstošā secībā, kā arī tiek apskatīta meklētāja vērtība un tiek meklēti uzdevumi kuros ir daļa no teksta, kas ierakstīta meklētājā. Pēc šiem kritērijiem dati tiek atlasīti un uzrādīti.



4.12. att. Uzdevumu attēlošanas datuplūsmas shēma

**10. Konkrēta uzdevuma attēlošana**

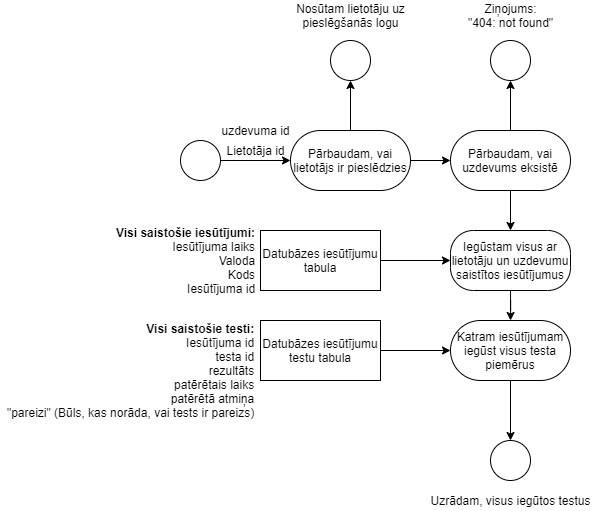
Nospiežot uz jebkuru uzdevumu kopējā uzdevumu meklētājā lietotājs tiek nosūtīts uz uzdevuma apskates lapu, kur tiek uzrādīta informācija par uzdevumu. Ja lietotājs ir pieslēdzies sistēmai tiek uzrādītas pogas, kas ļauj lietotājam izmantot sev pieejamo funkcionalitāti – iesūtīt uzdevumu, apskatīt savus iesūtījumus, un apskatīt savus atrisinājumus. Ja lietotājs ir pieslēdzies un lietotājam ir admina status, tad tiek uzrādītas pogas, kas dod admin funkcionalitāti – labot un dzēst uzdevumu.



4.13. att. Konkrēta uzdevuma attēlošanas datu plūsmas shēma

**11. Lietotāja iesūtījumu attēlošana**

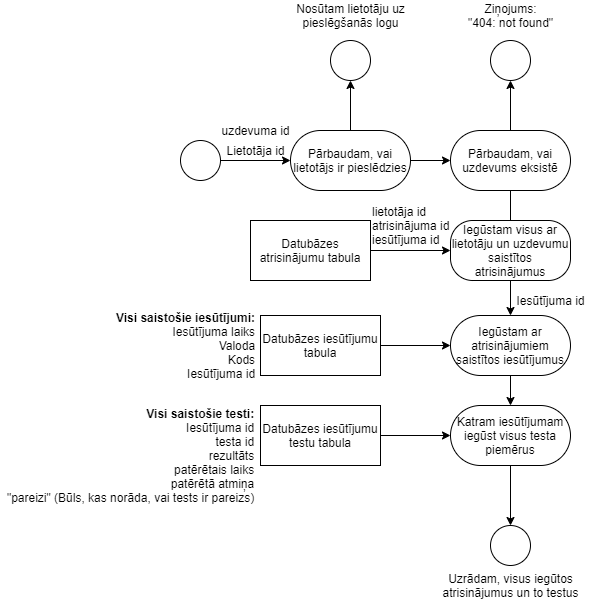
Ja lietotājs ir pieslēdzies, tam ļauj apskatīt konkrētam uzdevumam sevis veiktos iesūtījumus. No datubāzes tiek iegūti visi dati, kas ir piesaistīti šim lietotājam un uzrādīti mājslapā.



4.14. Lietotāja iesūtījumu attēlošanas datu plūsmas shēma

**12. Lietotāja atrisinājumu attēlošana**

Ja lietotājs ir pieslēdzies, tam ļauj apskatīt konkrētam uzdevumam savus veiksmīgos iesūtījumus, jeb atrisinājumus. No datubāzes tiek iegūti visi ieraksti, kas šim lietotājam ir pievienoti, kā veiksmīgi atrisinājumi un uzrādīti mājaslapā.



4.15. att Uzdevuma atrisinājumu attēlošanas datu plūsmas shēma

**13. Iesūtīt uzdevumu**

Galvenais sistēmas uzdevums ir ļaut lietotājiem iesūtīt savus programmēšānas uzdevumus un redzēt, vai uzdevumi ir atrisināti pareizi. Ja lietotājs ir pieslēdzies ir iespējams konkrētam uzdevumam veikt iesūtījumu. Lietotājam atveras logs kurā ir izvēles lauks, kur izvēlēties valodu (noklusējumā valoda ir C++) un lauks kurā rakstīt kodu. Balstoties uz valodu, kura izvēlēta koda redaktors iekrāsos attiecīgos programmas atslēgvārdus, un iekrāsos sintaksi.

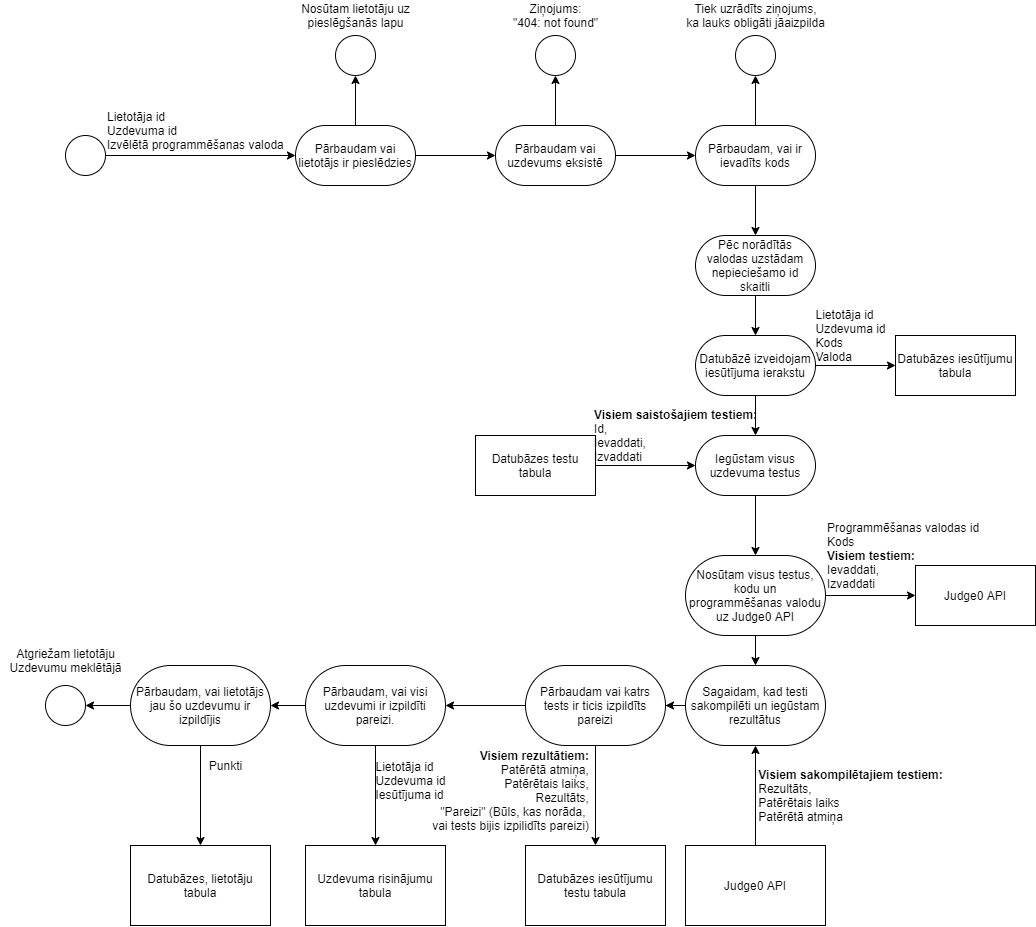
Ja koda lauks nav tukšs nospiežot pogu “sūtīt” informācija par kodu un izvēlēto valodu tiek sūtīta tālāk, ja lauks ir bijis tukšs, tad tiek uzrādīta ziņa, ka lauks ir obligati jāaizpilda.

Kad informācija nosūtīta uz kontrolieri, tiek pārbaudīts kura valoda tika izvēlēta un pēc šī izvēlās pareizo id, kas tiks izmantots iekš kompilatora API. Tiek izveidots iesūtījuma ieraksts datubāzē, kad tas izdarīts, visi testi tiek ievietoti masīvā un kopā ar programmēšanas valodas id, un uzdevuma ievaddatiem, ar HTTP request palīdzību, tiek nosūtīti uz Judge0 API.

Tiek sagaidīts, kad visi testi sakompilēti un tiek iegūti visu testu rezultāti – kādi izvaddati katriem ievaddatiem ir bijuši ar lietotāja ievadīto kodu. Kad rezultāti iegūti tie tiek pārbaudīti – tiek pārbaudīts, vai rezultāts:

1. Ietilpst laika limitā
2. Ietilpst atmiņas limitā
3. Faktiskie izvaddati sakrīt ar paredzētajiem izvaddatiem

Ja visi šie kritēriji ir ievēroti tests tiek atzīmēts kā korekts, ja nē, tad, ka nav izpildīts  
pareizi, bet jebkurā gadījumā tiek saglabāts datubāzē. Kad visi testi salīdzināti tiek pārbaudīts, vai uzdevums ir izpildīts pareizi – visi testi ir bijuši pareizi. Ja uzdevums ir izpildīts pareizi tiek izveidots “atrisinājuma” (solution) ieraksts. Ja šim lietotājam, šis uzdevums vēl nav bijis atrisināts (atrisinājuma ieraksta nav) lietotājam tiek piešķirti punkti par šī uzdevuma izpildīšanu.



4.16. att. Uzdevumu iesūtīšanas datu plūsmas shēma

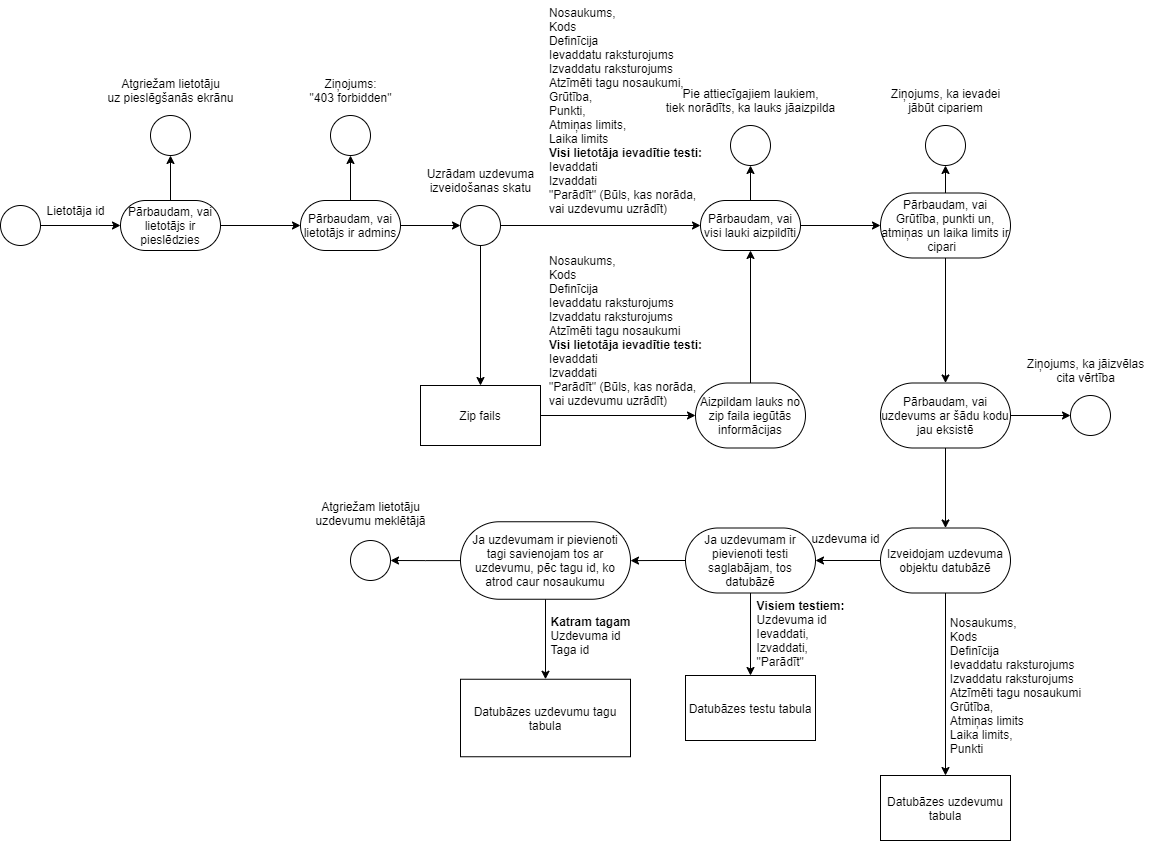
**14. Uzdevuma izveidošana**

Ja lietotājs ir pieslēdzies un tam ir admin statuss ir iespējams pievienot jaunu uzdevumu, nospiežot uz pogas “izveidot uzdevumu”, uzdevumu izvēlnes augšējā labajā stūrī. Lai izveidotu uzdevumu jāaizpilda forma. Tiek pārbaudīts, vai uzdevuma koda, uzdevuma nosaukuma, ievaddatu raksturojuma, izvaddatu raksturojuma, uzdevuma definīcjas, laika limita, atmiņas limita, punktu un grūtības lauki ir aizpildīti, ja nav pie attiecīgajiem laukiem, tiek uzrādīta ziņa, ka lauki jāaizpilda. Ir iespējams pievienot zip failu, kas izveidots, caur polygon sistēmu – ja zip fails korekti strukturizēts, lauki tiek aizpildīti automātiski (Grūtība, punkti, laika limits, atmiņas limits jāaizpilda pašam..

Tiek pārbaudīts, vai uzdevuma identifikācijas kods ir unikāls, ja nav, tiek uzrādīta ziņa, ka jāizvēlas cita vērtība. Ja šis kritērijs ievērots, tad datubāzē tiek izveidots jauns uzdevuma objekts ar ievadītajiem datiem.

Pievienot tagus nav obligāti, bet ja tie ir pievienoti, tiek pārbaudīts, vai kādi tagi ir tikuši pievienoti, ja ir, katrs tags tiek datubāzē sasaistīts ar uzdevumu ar daudzi pret daudziem saiti.

Pievienot testa piemērus nav obligāti, bet ja kāds ir pievienots, katram testam, kam ir gan ievaddati, gan izvaddati tiek izveidots ieraksts testu tabulā un sasaistīts ar uzdevumu.



4.17. att. Uzdevuma izveidošanas datu plūsmas shēma

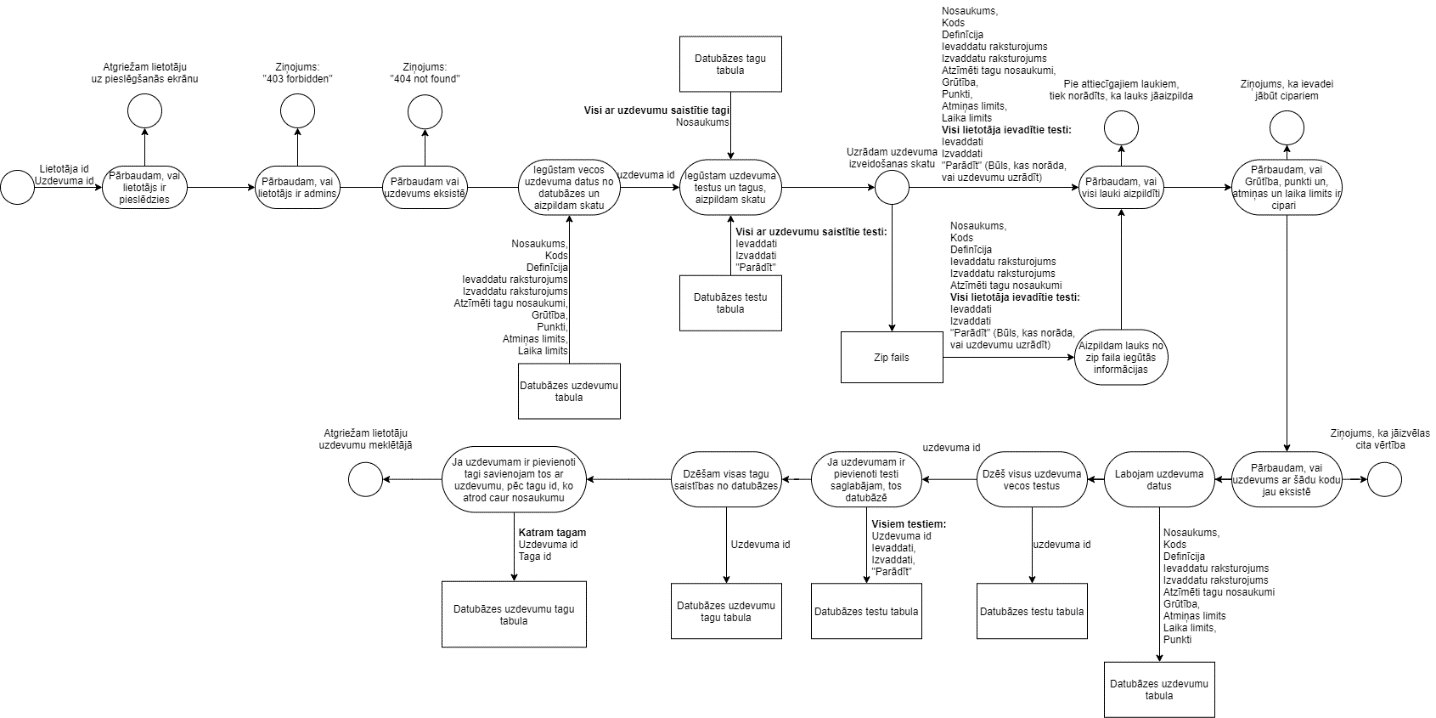
**15. Uzdevuma labošana**

Ja lietotājs ir pieslēdzies un tam ir admin statuss ir iespējams labot kādu uzdevumu, nospiežot uz pogas “Labot uzdevumu”, katra uzdevuma individuālajā skatā. Atverot uzdevuma labošanas skatu, tiek uzrādīta forma, kura jau aizpildīta ar uzdevuma vecajām vērtībām. Tiek pārbaudīts, vai uzdevuma koda, uzdevuma nosaukuma, ievaddatu raksturojuma, izvaddatu raksturojuma, uzdevuma definīcjas, laika limita, atmiņas limita, punktu un grūtības lauki ir aizpildīti, ja nav pie attiecīgajiem laukiem, tiek uzrādīta ziņa, ka lauki jāaizpilda.

Tiek pārbaudīts, vai uzdevuma identifikācijas kods ir unikāls, ja nav, tiek uzrādīta ziņa, ka jāizvēlas cita vērtība. Ja šis kritērijs ievērots, tad datubāzē tiek labots dotais uzdevuma objekts ar ievadītajiem datiem.

Uzdevuma labošanas brīdī visi tagi tiek atsaistīti no uzdevuma un tiek pievienots jaunais tagu saraksts. Pievienot tagus nav obligāti, bet ja tie ir pievienoti, tiek pārbaudīts, vai kādi tagi ir tikuši pievienoti, ja ir, katrs tags tiek datubāzē sasaistīts ar uzdevumu ar daudzi pret daudziem saiti.

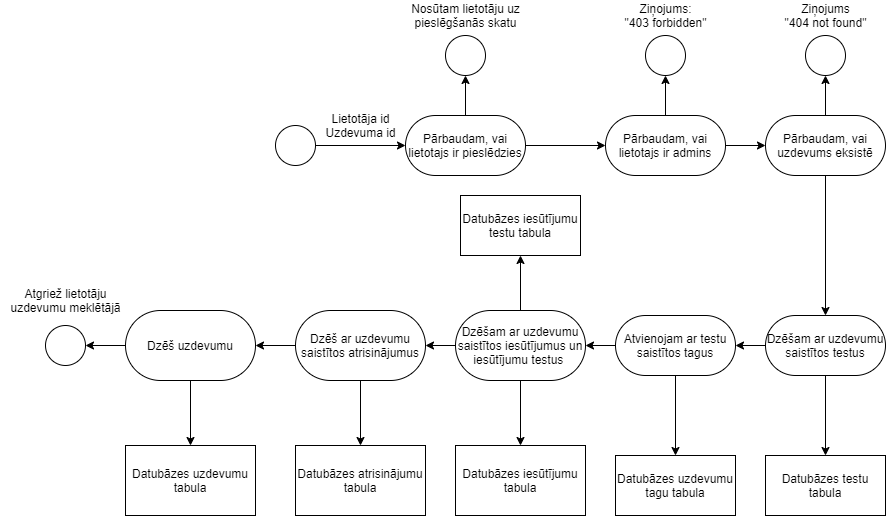
Uzdevuma labošanas brīdī, visi testi tiek dzēsti un tiek pievienots jaunais testu saraksts. Pievienot testapiemērus nav obligāti, bet ja kāds ir pievienots, katram testam, kam ir gan ievaddati, gan izvaddati tiek izveidots ieraksts testu tabulā un sasaistīts ar uzdevumu.



4.18. Uzdevumu labošanas datu plūsmas shēma

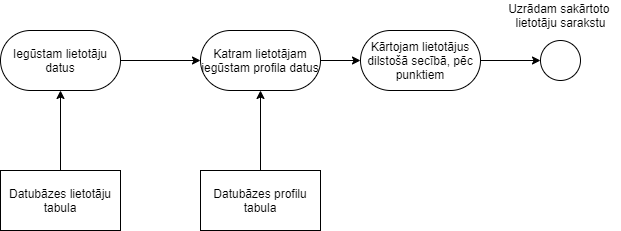
**16. Uzdevumu dzēšana**

Ja lietotājs ir pieslēdzies un tam ir admina statuss, pie individuālā uzdevuma skata, nospiežot uz pogas “Dzēst uzdevumu”, nospiežot uz šīs pogas no datubāzes tiek dzēsts dotais uzdevums.



14.19. att. Uzdevumu dzēšana datu plūsmas shēma

**17. Reitinga attēlošana**

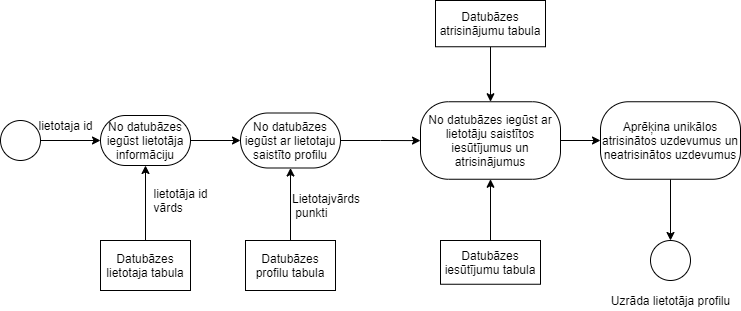
Reitinga skatā, tiek attēlots lietotāju reitings pēc punktiem. No datubāzes tiek iegūti visi lietotāju ieraksti, kas tiek sakārtoti pēc iegūto punktu daudzuma un uzrādīti sarakstā. 

14.20. Reitinga attēlošanas datu plūsmas shēma

**18. Profila attēlošana**

Ir iespējams apskatīt jebkura lietotāja profila statistiku. No datubāzes tiek iegūta lietotāja informācija un visi ar lietotāju saistītie iesūtījumi un atrisinājumi.Tiek uzrādīta lietotāja statistika – lietotajvārds, punkti, tiek aprēķināts uzdevumu reitings, skaits ar atrisinātiem uzdevumiem, skaits ar neatrisinātiem uzdevumiem. Kopējo atrisinājumu skaitu (arī dublikātus) uzrāda vizuālā grafikā.

Tiek atlasīti uzdevumu, kurus lietotājs ir izpildījis un uzdevumus, kurus lietotājs mēģināja pildīt, bet vēl nav atrisinājis. Šie dati tiek ievietoti uzdevumu meklētāja VUE.js komponentā.



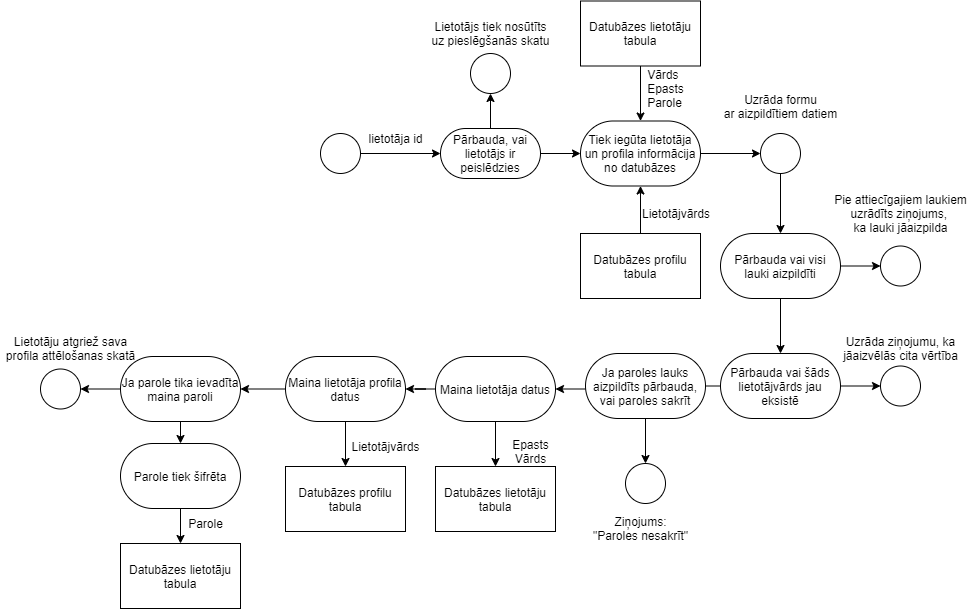
4.21. att Lietotāja profila attēlošanas shēma

**19. Profila labošana**

Ja lietotājs ir pieslēdzies ir pieejams labot savus profila datus, ja nav lietotājs tiek nosūtīts uz pieslēgšanās lapu. Lietotāja informācijas labošanas skatā tiek uzrādīta forma ar vecajiem informācijas datiem – lietotājvārdu, vārdu, un epastu, kā arī neaizpildīti lauki – lauks “parole” un lauks “atkārtot paroli”. Laukiem lietotājvārds, vārds un epasts obligāti ir jābūt aizpildītiem, ja tie nebūs, tad tiks uzrādīts ziņojums, ka laukiem ir jābūt aizpildītiem.

Paroles laukiem nav obligāti jābūt aizpildītiem, bet ja lauks “parole” ir aizpildīts, laukā “atkārtot paroli” jābūt tādai pašai parolei (lauks nevar būt tukšs), citādāk tiks uzrādīts ziņojums, ka paroles nesakrīt.

Ja visi šie kritēriji ir ievēroti, tad pieslēgtā lietotāja informācija tiek mainīta uz jauno informāciju.



4.22. att. Profila labošanas datu plūsmas shēma

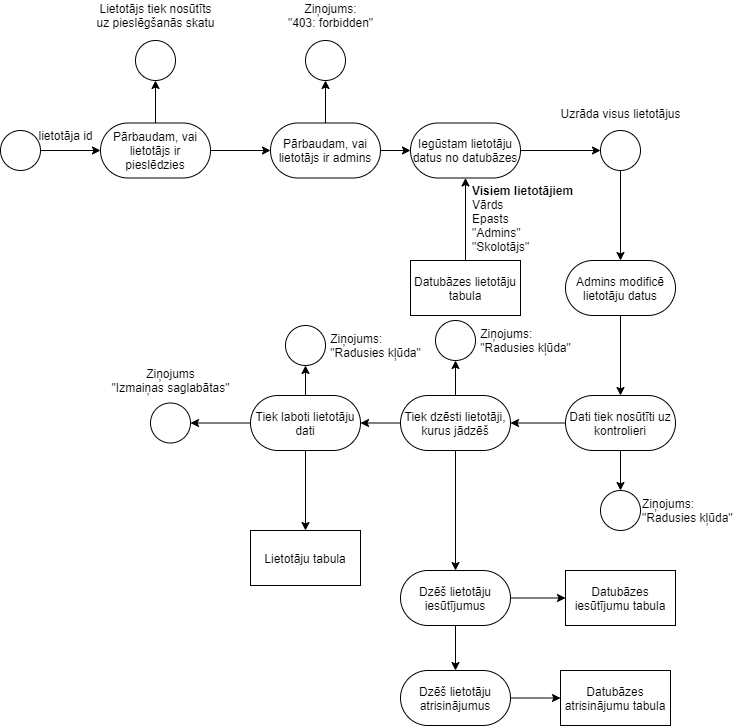
**20. Lietotāju rediģēšana**

Ja lietotājs ir pieslēdzies un tam ir admin statuss, ir iespējams labot lietotāju privilēģijas.

Lietotāju rediģēšanas skatā tiek iegūta visu lietotāju informācija un tiek saraksta veidā uzrādīts katra lietotāja vārds, epasts un atzīmējami lauki, kas norāda uz lietotāja admina statusu un skolotāja statusu – ja statuss ir piešķirts lauks ir atzīmēts. Pie katra lietotāja ir arī dzēšanas poga, kas noņem lietotāju no kopējā saraksta.

Izmaiņas datubāzē tiek saglabātas, kad nospiesta poga “Saglabāt izmaiņas”. Kad poga nospiesta uz kontrolieri tiek nosūtīti lietotāji, kas jāizdzēš un lietotāji, kuriem jāmaina statusi.

Ja lietotājs jādzēš tas tiek izdzēsts, ja jāmaina statusi lietotājiem tiek mainīti attiecīgie statusi.



4.23. att. Lietotāju rediģēšanas datu plūsmas shēma

**21. Tagu rediģēšana**

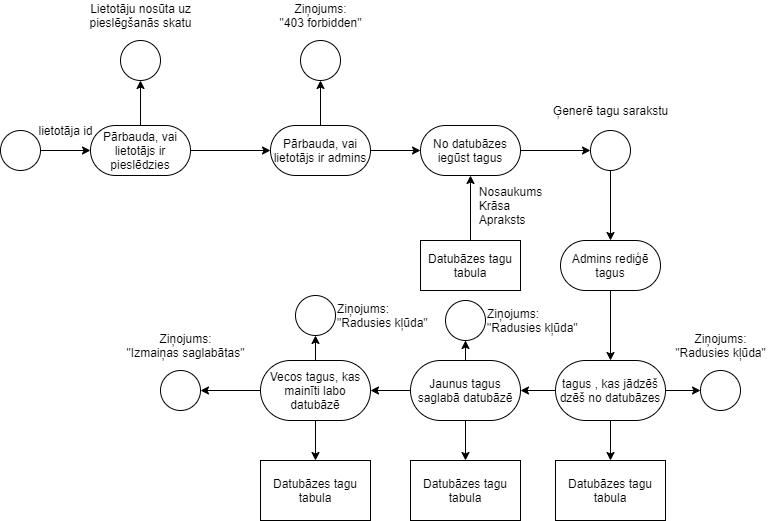
Ja lietotājs ir pieslēdzies un tam ir admin statuss, ir iespējams rediģēt tagus.

Ieejot tagu rediģēšanas skatā tiek iegūti visi jau esošie tagi, no datubāzes un ievietoti ievadlaukos, taga izvēlētā krāsa, tiek ievietota selektorā, kā arī ievadlauku rindas fona krāsa ir taga krāsa.

Ja tags tiek mainīts, tas tiek atzīmēts kā labojams, ja tags tiek dzēsts tas tiek atzīmēts kā dzēšams, tagi, kas ir tikko pievienoti, tiek atzīmēti, kā jauni.

Tagi tiks pievienoti, tikai tad ja lauks “nosaukums” tiks aizpildīts, ja lauks būš tukšs tags tiks ignorēts.

Tagu objekti tiek nosūtīti uz kontrolieri un pēc iepriekš piešķirtajiem apzīmējumiem dzēšamie tagi tiek dzēsti, labojamie tiek laboti un jaunie tagi tiek izveidoti datubāzē, kā jauni objekti.



4.23. att Tagu rediģēšanas sistēmas datu plūsmas shēma

# 5. DATU STRUKTŪRAS APRAKSTS

Datubāze sastāv no 11 tabulām, kas satur informāciju par sistēmas lietotājiem, uzdevumiem, iesūtījumiem, kā arī lietotājiem redzamajiem “Jaunumu” ziņojumiem un reklāmas bildēm. “Exercise” ir galvenais platformas objekts, kas savienojas ar vairākām tabulām. Datubāzes projektēšanas procesā, lai tabulas atbilstu realizējamiem uzdevumiem un tiktu novērsta datu dublēšanās, tiek veikta to vienkāršošana un optimizēšana jeb normalizācija. 2.pielikumā dotājā attēlā ir parādīta datu bāzes struktūra, kas demonstrē datu kolonu izvietojumu pa tabulām, kā arī tabulu savstarpējo saistību:

Tabulā “**Exercise**” glabājas informācija par izveidotajiem uzdevumiem (Uzdevumus var izveidot tikai admins). Informācija par uzdevumu, t.sk. nosaukumu, identifikācijas kodu,, uzdevuma definīciju, punktu skaitu, grūtības pakāpi, kā arī maksimāli pieļaujamo uzdevuma izpildes laiku un aizņemamo atmiņu.

5.1. tabula

**1. “Exercise” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | nosaukums | string | 8000 | uzdevuma nosaukums |
| 3 | definicija | text | 8000 | uzdevuma apraksts |
| 4 | score | int | 2^31 | iegūstamie punkti |
| 5 | difficulty | int | 2^31 | grūtības pakāpe |
| 6 | time | float | 1000 | pieļaujamais laiks (sek) |
| 7 | memory | float | 1000 | pieļaujamā atmiņa (MB) |
| 8 | kods | string | 200 | identifikācijas kods |

Tabulā “**Test**” glabājas informācija par uzdevuma testiem - ievaddati, izvaddati un buleāns, kas norāda uz to, vai testapiemēru var publiski apskatīt. Tabula savienojas ar tabulu “Exercise”, jo vienam uzdevumam, varētu būt vairāki testi.

5.2. tabula

**2. “Test” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | exercise\_id | int | - | FK, kas norāda uz uzdevumu |
| 3 | stdout | text | 8000 | sagaidāmie izvaddati |
| 4 | stdin | text | 8000 | testa ievaddati |

Tabulā “**Tag**” glabājas informācija par uzdevumu tagiem - taga nosaukums, apraksts un krāsa. Tabula ar starp-galdiņa. Tabula ar starp-galdiņa palīdzību savienojas ar “Exercises”, jo vienam uzdevumam varētu būt vairāki tagi un viens tags var būt vairākiem uzdevumiem.

5.3. tabula

**3. “Tag” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | name | string | 8000 | taga nosaukums |
| 3 | desc | string | 8000 | taga apraksts |
| 4 | color | string | 8000 | krāsas heksadecimālā vērtība |

Tabulā “**Submission**” glabājas informācija visu uzdevumu atsevišķajiem iesūtījumiem - kods un valoda kurā tas ir iesūtīts. Tabula savienojas ar “Exercise”, jo vienam uzdevumam var būt vairāki iesūtījumi.

5.4. tabula

**4. “Submission” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | exercise\_id | int | - | FK, kas norāda uz iesūtīto uzdevumu |
| 3 | user\_id | int | - | FK, kas norāda uz lietotāju, kas uzdevumu iesūtīja |
| 4 | code | text | 8000 | iesūtījuma kods |
| 5 | mode | string | 8000 | iesūtījuma programmēšanas valoda |

Tabulā “**SubmissionTest**” glabājas informācija visu katru uzdevuma atseviškiem testa piemēriem - laiks, atmiņa, ko ir patērējis uzdevumas, izvaddati un buleāns, kas norāda vai tests ir izpildīts pareizi.. Tabula savienojas ar “Submission”, jo vienam iesūtījumam var būt vairāki testi.

5.5. tabula

**5. “SubmissionTest” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | submission\_id | int | - | FK, kas norāda uz iesūtījumu |
| 3 | time | string | - | laiks, ko patērējis tests |
| 4 | memory | float | 1000 | atmiņa, ko patērējis tests |
| 5 | correct | boolean | - | norāda vai tests ir izpildīts korekti |

Tabulā “**Solution**” glabājas informācija pareizi atrisinātajiem uzdevumiem - tabulai nav savu datu, tikai FK, kas norāda uz pareizo iesūtījumu.

5.6. tabula

**6. “Solution” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | submission\_id | int | - | FK, kas norāda uz iesūtījumu |
| 3 | user\_id | int | - | FK, kas norāda uz lietotāju |
| 4 | exercise\_id | int | - | FK, kas norāda uz uzdevumu |

Tabulā “**User**” glabājas publiski nepieejamā informācija par lietotājiem - vārds, epasts, parole (aizkodēta ar hash), booleāns, kas norāda vai lietotājs ir admins un buleāns, kas norāda vai lietotājs ir skolotajs.

5.7. tabula

**7. “User” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | name | string | - | Lietotāja vārds |
| 3 | email | string | - | Lietotāja epasts |
| 4 | password | string | - | Lietotāja parole |
| 5 | admin | boolean | - | Norāda uz admina statusu |
| 6 | teacher | boolean | - | Norāda uz skolotāja statusu |

Tabulā “**Profile**” glabājas publiski pieejamā informācija par lietotājiem - punktu skaits un lietotājvārds. Savienots ar 1-1 saiti ar “User” tabulu , jo katram lietotājam ir tieši viens profils

5.8. tabula

**8. “Profile” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | user\_id | int | - | FK, kas norāda uz lietotāju |
| 3 | score | int | 2^31 | Lietotāja kopējais punktu skaits |
| 4 | username | string | - | Lietotāja lietotājvārds |

Tabulā “**News**” glabājas ziņu ieraksti - nosaukums un apraksts. Savienots ar ar “User” tabulu , lai varētu redzēt kurš admins ievietojis ziņojumu.

5.9. tabula

**9. “News” tabulas struktūra**

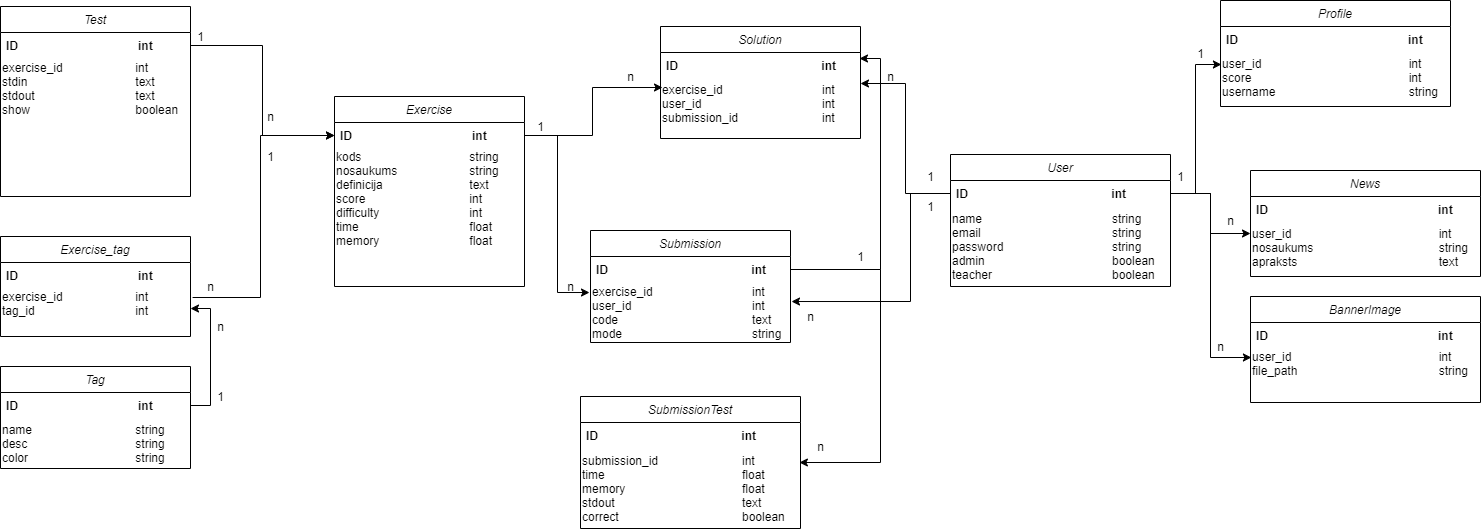
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | user\_id | int | - | FK, kas norāda uz lietotāju |
| 3 | nosaukums | text | 8000 | Ziņojuma nosaukums |
| 4 | apraksts | text | 8000 | Ziņojuma apraksts |

Tabulā “**BannerImages**” glabājas informācija par galvenā banera bildēm - ceļš, kur atrodas bilde. Savienots ar ar “User” tabulu , lai varētu redzēt kurš admins ievietojis bildi.

5.10. tabula

**10. “BannerImages” tabulas struktūra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīmes** |
| 1 | id | autonumber | - | Primārā atslēga |
| 2 | user\_id | int | - | FK, kas norāda uz lietotāju |
| 3 | file\_path | text | 8000 | ceļš uz banera bildi |



4.23. att. Datubāzes struktūras shēma

# 6. LIETOTĀJA CEĻVEDIS

## 6.1. Sistēmas prasības aparatūrai un programmatūrai

Lai sāktu darbu ar sistēmu to ir nepieciešams uzstādīt uz servera. Sistēma izmanto Judge0 API kompilatoru – bez šī API pieslēguma sistēma strādās kļūdaini. Judge0 ir “open source” programmatūra, to var lejupielādēt un uzstādīt uz sava servera, vai arī izmantot oficiālo Judge0 hostu.

Mājaslapa ir būvēta izmantojot AdminLTE veidni (template), interneta pārlūki un to versijas, dokumentācijas rakstīšanas laikā – jaunākās versijas ko sistēma atbalsta:

* Internet explorer 9+
* Edge (v46.3)
* Firefox (v89)
* Safari (v10.11)
* Chrome (v91)
* Opera (v76)

Veidne vēljoprojām tiek atbalstīta un atjaunota, ja sistēma tiek pareizi uzturēta, tuvāko gadu laikā tai vajadzētu strādāt ar gandrīz jebkuru no iepriekšminētajiem interneta pārlūkiem, ar citām pārlūkprogrammām sistēma var strādāt nestabili. Ieteicamā operētājsistēma, būtu kāda, kas atbalsta kādu no minētajiem interneta pārlūkiem.

Mājaslapu uzturēšana ar mūsidienu tehnoloģiju nav problēma, par cik šāda sistēma nav prasīga, tai visticamāk, derētu jebkāds serveris – ieteicamie datortehnikas parametri būtu:

* RAM – 1GB
* CPU – 500Mhz
* Vieta cietajā diskā – 10GB (Pieņemot, ka tiks glabātas augstas kvalitātes bildes)

Programmatūras prasības serverim – sistēmai ir jāievēro ietvara prasības, lai Laravel ietvars korekti darbotos (Šīs ir oficiālās laravel izstrādātāju prasības):

* PHP >= 5.4, PHP <7
* Mcrypt PHP paplašinājums
* OpenSSL PHP paplašinājums
* Mbstring PHP paplašinājums
* Tokenizer PHP paplašinājums

Serverim uz kura sistēma tiks uztādīta ir nepieciešams HTTP serveris, lai sistēma varētu pieslēgties internetam. Ieteicams izmantot vispopulārākos serverus, kā piemēram apache vai nginx.

Lai uzstādītu programmatūru ir nepieciešams Composer “dependency manager”, bez šī programmas uzstādīšana nebūs iespējama – tas tiek izmantots, lai instalētu laravel ietvaru un tā funkcionalitāti. Nepieciešams arī Node.js, lai uzstādītu spraudņus un citas “dependency”

Ļoti ieteicams uzstādīt kādu servera menedžmenta programmu, kā piemēram cPanel, lai uzstādīšanas un uzturēšanas procesi būtu ērtāki.

Lai uzstādītu sistēmu ir nepieciešams serverim dot komandas, kas ļaus composer programmatūrai uzstādīt laravel ietvaru – šim ieteicams izmantot SSH termināli, piemēram PuTTY.

Uz servera jābūt datubāžu vadības sistēmai MySQL (versiju 8) tieši uz šīs versijas tika veidota sistēma.

Ieteicams uzstādīt datubāžu pārvaldes rīku phpMyAdmin – šī sistēma nav obligāta, bet palīdz ļoti ērti pārvaldīt datubāzes.

## 6.2. Sistēmas instalācija un palaišana

Visa sistēmas funkcionalitāte – kā admina, tā lietotāja – atrodas vienā un tajā pašā sistēmā, bet sistēmas logi, izkārtojums, un funkcionalitātes mainīsies atkarīgi no lietotāja privilēģijām. Tas nozīmē, ka sistēmai jāveic tikai viens uzstādīšanas process un sistēma būs gatava darbam.

* **Uzstādīšana testa vidē uz lokālas mašīnas**

1. Laravel ietvaram ir iebūvēts testa serveris, tāpēc sistēmu testēšanai var uzstādīt arī bez HTTP servera. Kad faili lejupielādēti caur terminālī (Piem. Windows Cmd) palaist komandu “composer install”. (Šo un nākamo komandu laikā terminālī jāatrodas sistēmas atrašanās vietā). Šī komanda instalēs un uzstādīs laravel ietvaru.
2. Terminālī palaist komandu “npm install” šī komanda uzstādīs visu nepieciešamo, priekš Node.js.
3. Galvenajā sistēmas datnē jāatrodas failam “.env.example” kopēt un pārdēvēt šo failu par “.env” (Izmantojot komandu, tas būtu “cp .env.example .env”
4. Sistēmas database datnē izveidot failu “database.sqlite”
5. .env failā dzēst datubāzes parametrus (10. – 15. rindiņas). To vietā norādīt “DB\_CONNECTION=sqlite”
6. .env failā QUEUE\_CONNECTION=sync (19. rindiņa) noaminīt uz QUEUE\_CONNECTION=database. Šis ļaus mājaslapai strādāt ar queue fona procesiem.
7. Palaist komandu: “php artisan key:generate” – ģenerē aplikācijas šifrēšanas atslēgu.
8. Palaist komandu: “php artisan migrate” – izveido pareizu datubāzes struktūru.
9. Palaist komandu “php artisan storage:link” – izveido simbolisko linku starp failu datnēm un laravel ietvaru.
10. Lai palaistu pašu serveri terminālī palaist komandu “php artisan serve”
11. Lai palaistu fona procesus terminālī palaist komandu “php artisan queue:work”

* **Uzstādīšana uz HTTP severa**

1. Sistēmas faili ir jāuzstāda uz servera. Dažādiem serveriem datņu struktūras atšķirās, tāpēc kur konkrēti faili atradīsies ir atkarīgs no situācijas. Ja sistēmu uzstāda uz subdomēna failiem ir jāatrodas “public\_html/subdomena\_vards/public” datnē, jo laravel ietvaram ir paredzēts, ka pamata faili atrodas “public” datnē.
2. Jāuzstāda datubāze un lietotājs ar privilēģijām datubāzes ierakstus mainīt.
3. Serverī izveidotajā datnē jāaugšupielādē sistēmas faili.
4. Kad faili augšupielādēti caur SSH terminālī palaist komandu “composer install”. (Šo un nākamo komandu laikā SSH terminālī jāatrodas sistēmas atrašanās vietā). Šī komanda instalēs un uzstādīs laravel ietvaru.
5. Terminālī palaist komandu “npm install” šī komanda uzstādīs visu nepieciešamo, priekš Node.js.
6. Galvenajā sistēmas datnē jāatrodas failam “.env.example” kopēt un pārdēvēt šo failu par “.env” (Izmantojot komandu, tas būtu “cp .env.example .env”
7. .env failā uzstādīt datubāzes parametrus balstoties uz iepriekš izveidoto datubāzi un lietotāju (10. – 15. rindiņas)
8. .env failā QUEUE\_CONNECTION=sync (19. rindiņa) noaminīt uz QUEUE\_CONNECTION=database. Šis ļaus mājaslapai strādāt ar queue fona procesiem.
9. Palaist komandu: “php artisan key:generate” – ģenerē aplikācijas šifrēšanas atslēgu.
10. Palaist komandu: “php artisan migrate” – izveido pareizu datubāzes struktūru.
11. Palaist komandu “php artisan storage:link” – izveido simbolisko linku starp failu datnēm un laravel ietvaru.

* **Uzstādīt admin profilu –** Neatkarīgi no uzstādīšanas sistēmas caur SSH termināli jāuzstāda admin profils.
  1. Izveidot profilu reģistrējoties sistēmā, nospiežot uz pogas “reģistrēties” - sistēmai šobrīd vajadzētu pilnībā strādāt, bet admin privilēģijas nebūs pieejamas, jo lietotājiiem, kas tiks izveidoti netiks automātiski pievienots admina statuss.
  2. Caur termināli palaist komandu “php artisan tinker” šis mums atļaus mainīt datubāzes informāciju iekš laravel ietvara.
  3. Iegūt un saglabāt lietotāju mainīgajā: “$user = User::Where(‘id’,1)->first();”
  4. Mainīt objekta admin parametru: “$user->admin = 1;”
  5. Saglabāt izmaiņas: “$user->save();”

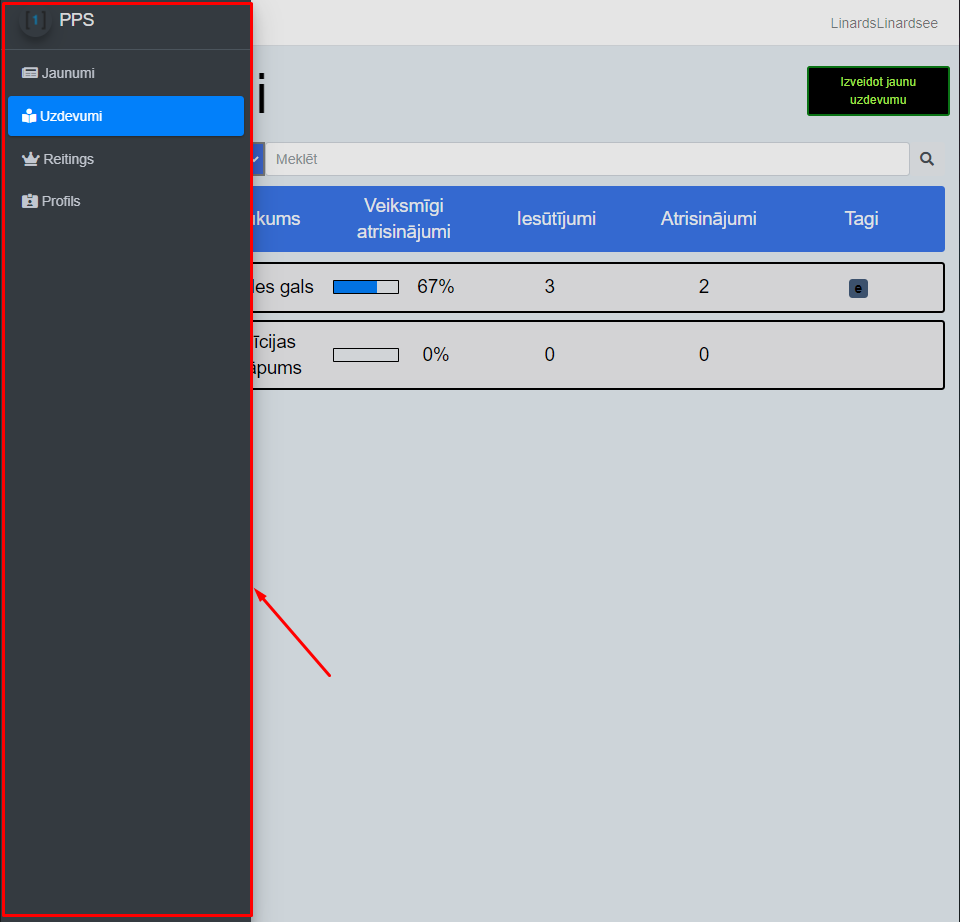
## 6.3. Programmas apraksts

Mājaslapas dizainā ir iekļautas divas navigācijas izvēlnes, kuras dinamiski pielāgojas ekrāna izmēram un izkārtojumam. Aizverama galvenā navigācijas izvēlne, kas atrodas mājaslapas kreisajā malā un lietotāja izvēlne, kas tiek uzrādīta nospiežotu uz sava lietotājvārda lapas augšējā, labajā stūrī.

Lapas galvenē atrodas navigācijas izvēlnes aizvēršanas/atvēršanas poga un lietotājvārds, kas kalpo kā poga, kas atver lietotāja izvēlni. Lapas ķermeņa daļā atrodas mājaslapas saturs, kas mainās atkarībā no tā, kurā skatā lietotājs atrodas.

**Navigācijas izvēlne**

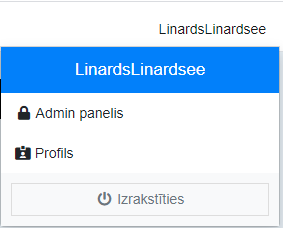
Kreisajā malā atrodas aizverama navigācijas izvēlne, šī izvēlne atrodas katrā skatā. Izvēlne lietotāju aizved uz mājaslapas galvenajām sadaļām – Jaunumu skatu, Uzdevumu skatu, reitinga skatu, vai, ja lietotājs ir pieslēdzies, uz sava profila skatu, ja lietotājs nav pieslēdzies poga, kas lietotāju ved uz profila skatu neuzrādās. Ja lietotājs atrodas kādā, no galvenajiem skatiem izvēlnē, tiek iekrāsots navigācijas links kurš ved uz doto skatu.



6.1. att. Atvērta navigācijas izvēlne, ar pieslēgušos lietotāju, kas atrodas uzdevumu skatā.

**Lietotāja izvēlne**

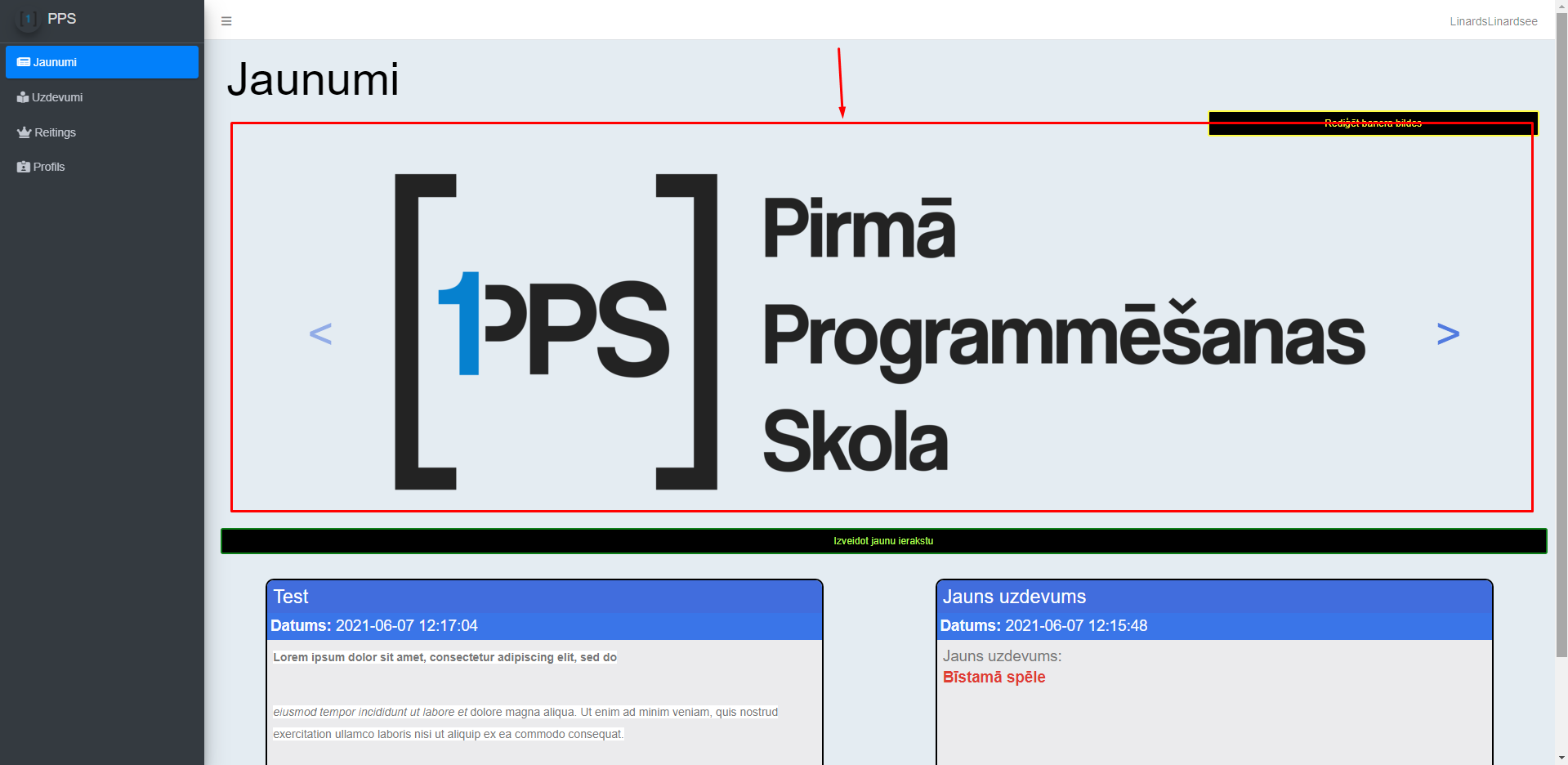
Lietotāja izvēlne tiek uzrādīta nospiežot uz sava lietotājvārda, kas atrodas lapas galvenē. Lietotāja izvēlnē ir pieejams nokļūt uz savu profilu, kā arī admin paneli, ja lietotājs ir admins, un izrakstīties no sistēmas, tādējādi beidzot savu sesiju.



6.2. arr. Atvērta lietotāja izvēlne lietotājam ar admin profilu

**Bilžu baneris**

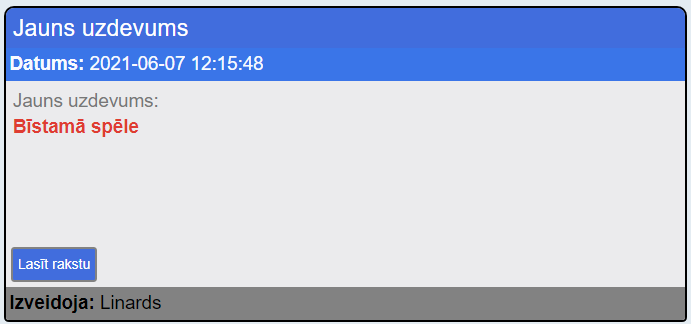
Ienākot mājaslapā pirmais, kur lietotājs iekļūst ir “Jaunumu” skats, kura augšējā daļā tiek uzrādīts bilžu baneris. Tas ir paredzēts, lai pievienotu, reklāmas bildes, vai ziņojumus bildes formātā.



6.3. att. Jaunumu skats ar apzīmētu bildes baneri

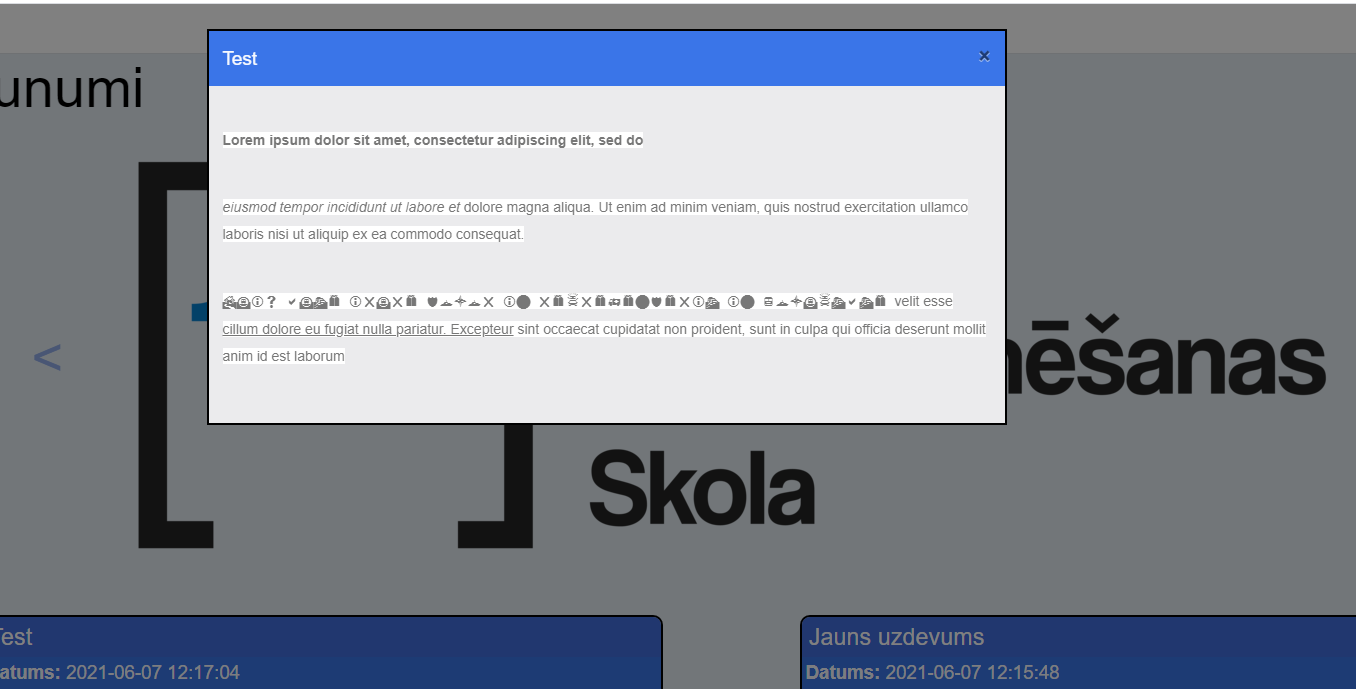
**Jaunumu ieraksti**

Otra jaunumu skata daļa sastāv no jaunumu ierakstiem – ziņojumiem teksta formātā, ko izveido admin lietotāji. Paredzēts, lai ziņotu par izmaiņām mājaslapas darbībā, reklāmu, vai jebkādas citas informācijas izplatīšanu, kas varētu interesēt lietotāju.



6.4. att.Ziņojuma kartiņas paraugs

Var rasties situācijas, ka teksts neietilpst ziņojuma kartē, tāpēc tas tiek apgriezts, nospiežot uz pogas “Lasīt rakstu” tiek atvērts modulis ar pilno tekstu – tests kā priekšskatījuma kartiņā, tā arī modulī var tikt noformēts ar dažādiem teksta efektiem, fontiem un stiliem (skat att 6.5).



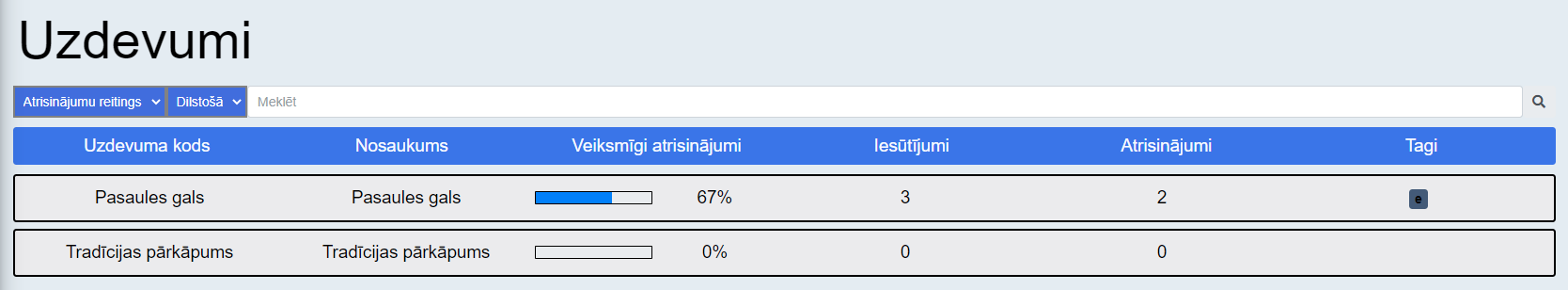
6.5. att. Paraugs ar moduli, kas uzrāda pilno ziņojuma testu

**Uzdevumu meklētājs**

Lietotājam ir nepieciešams ērti pārskatīt un meklēt visus uzdevumus. Uzdevumu meklētājs sastāv no 4 daļām:

* Filtrs, pēc kā kārtot uzdevumus
* Secības filtrs – kādā secībā uzrādīt uzdevumus
* Teksta meklētājs – kādam tekstam jābūt iekļautam uzdevuma nosaukumā
* Uzdevumu kopa – pēc kritērijiem tiek atlasīti visi uzdevumi un uzrādīti saraksta formā.

Lietotājam ir dinamiski (nepārlādējot lapu) mainīt kritērijus un uzdevumu saraksts pielāgosies pēc vajadzības.



6.6. att. Uzdevumi tiek kārtoti dilstošā secībā, pēc to veiksmīgo atrisinājumu reitinga

**Konkrēta uzdevuma attēlošana**

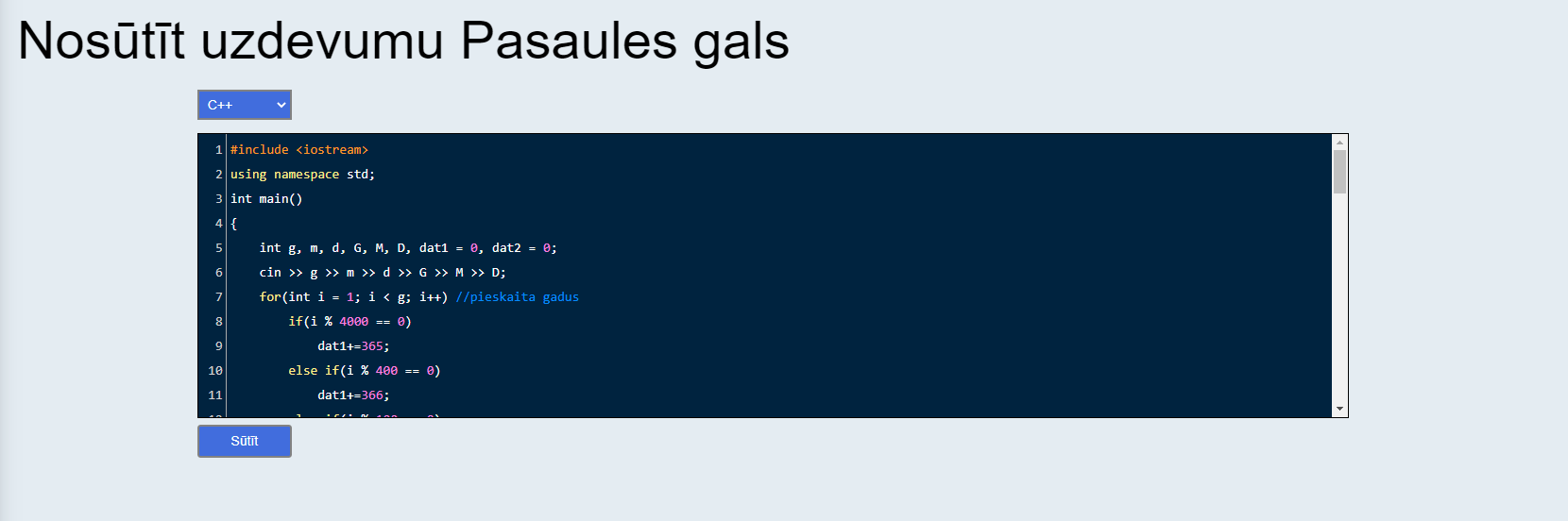
Katru uzdevumu ir iespējams attēlot individuāli, lai redzētu visu ar uzdevumu saistošo informāciju, kā piemēram laika un atmiņas limitus, paša uzdevumu prasības u.c. Atkarībā no tā vai lietotājs ir pieslēdzies tam uzrādās pogas: “Iesūtīt uzdevumu”, “Mani iesūtījumi”, “Mani atrisinājumi”. Ja lietotājs ir admins, tad uzrādās arī pogas “Dzēst uzdevumu” un “Labot uzdevumu”



6.7. att. Uzdevuma “Pasaules gals” attēlošana, kur sistemā pieslēdzies lietotājs ar admin privilēģijām.

**Uzdevuma iesūtīšana**

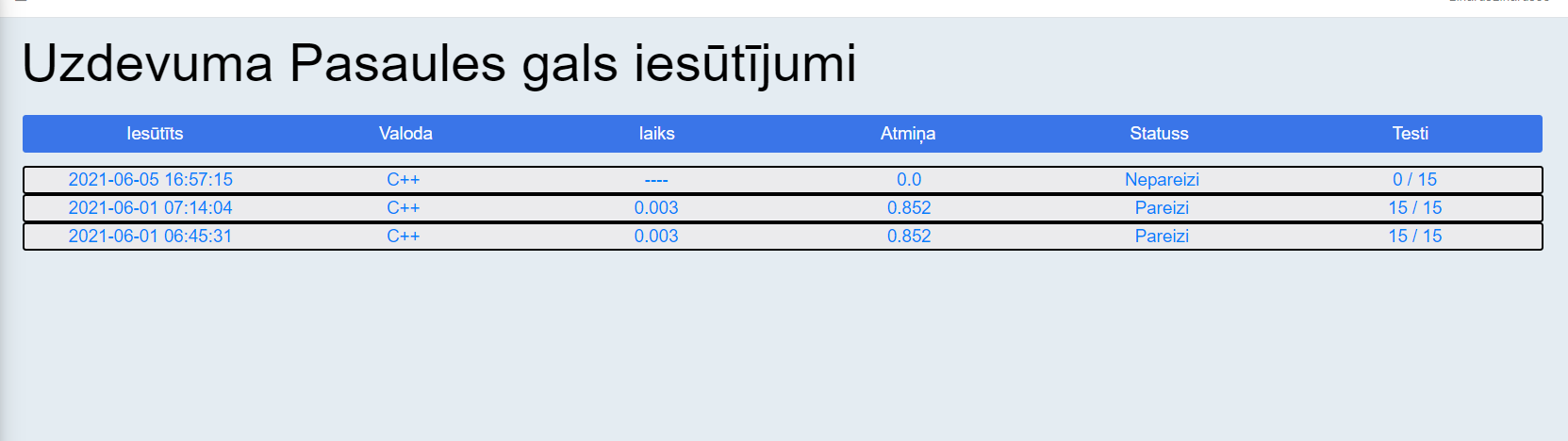
Lai uzdevumu izpildītu, lietotājam uzdevums ir jāiesūta. Iesūtīšanas logs sastāv no divām daļām – programmēšanas valodas izvēlnes un CodeMirror koda redaktora, kur rakstīt kodu. Atrisinājumu skats izskatās identiski, vienīgais, kas atšķirās ir atlasītie dati.



6.8. att. Uzdevuma iesūtīšanas logs ar aizpildītu koda lauku un izvēlēto valodu C++

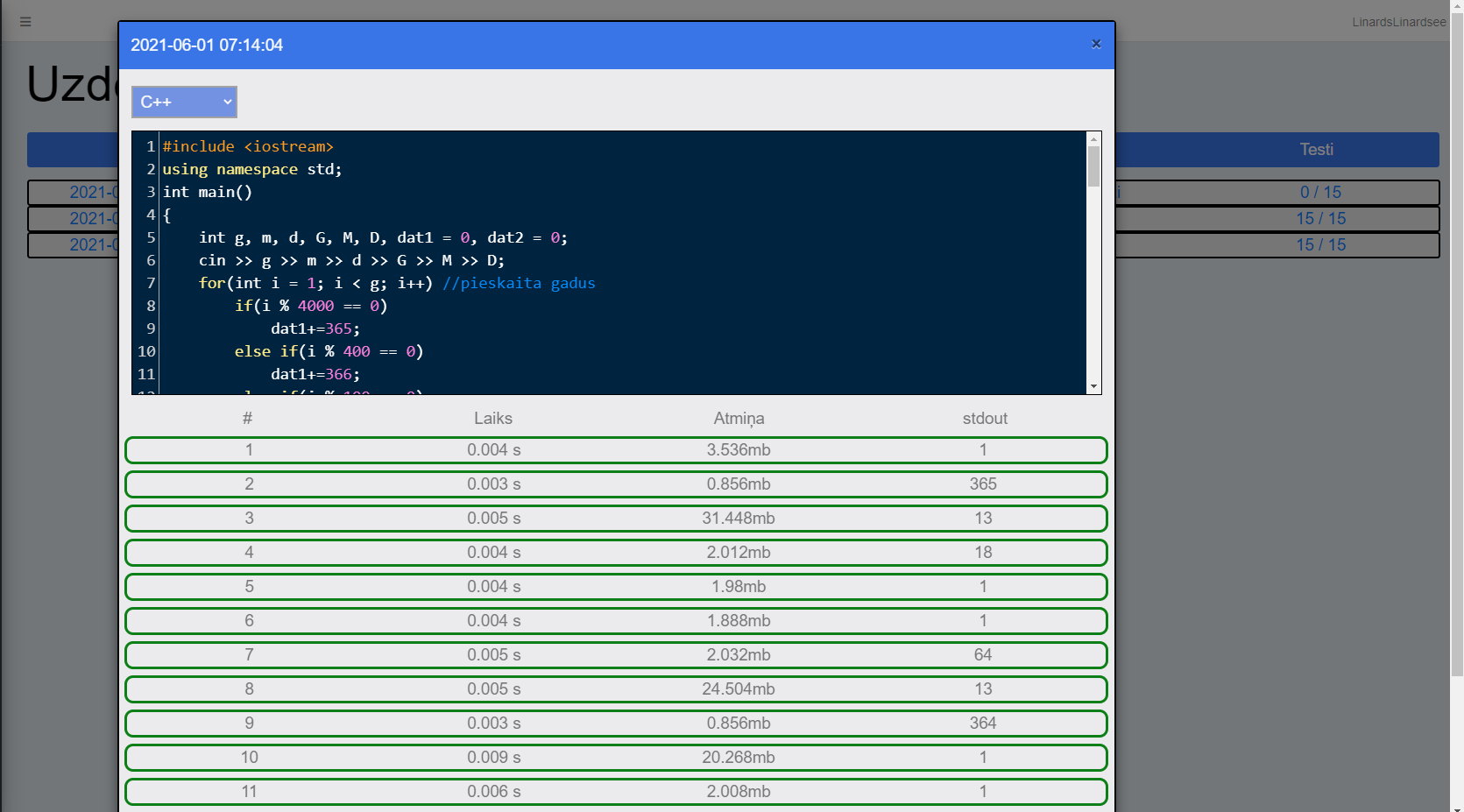
**Iesūtījumu apskatīšana**

Lai apskatītu savus vecos iesūtījumus un to rezutlātus, ir iespējams pie katra uzdevuma apskatīt iesūtījumu sarakstu.



6.9. att. Iesūtījumu saraksts

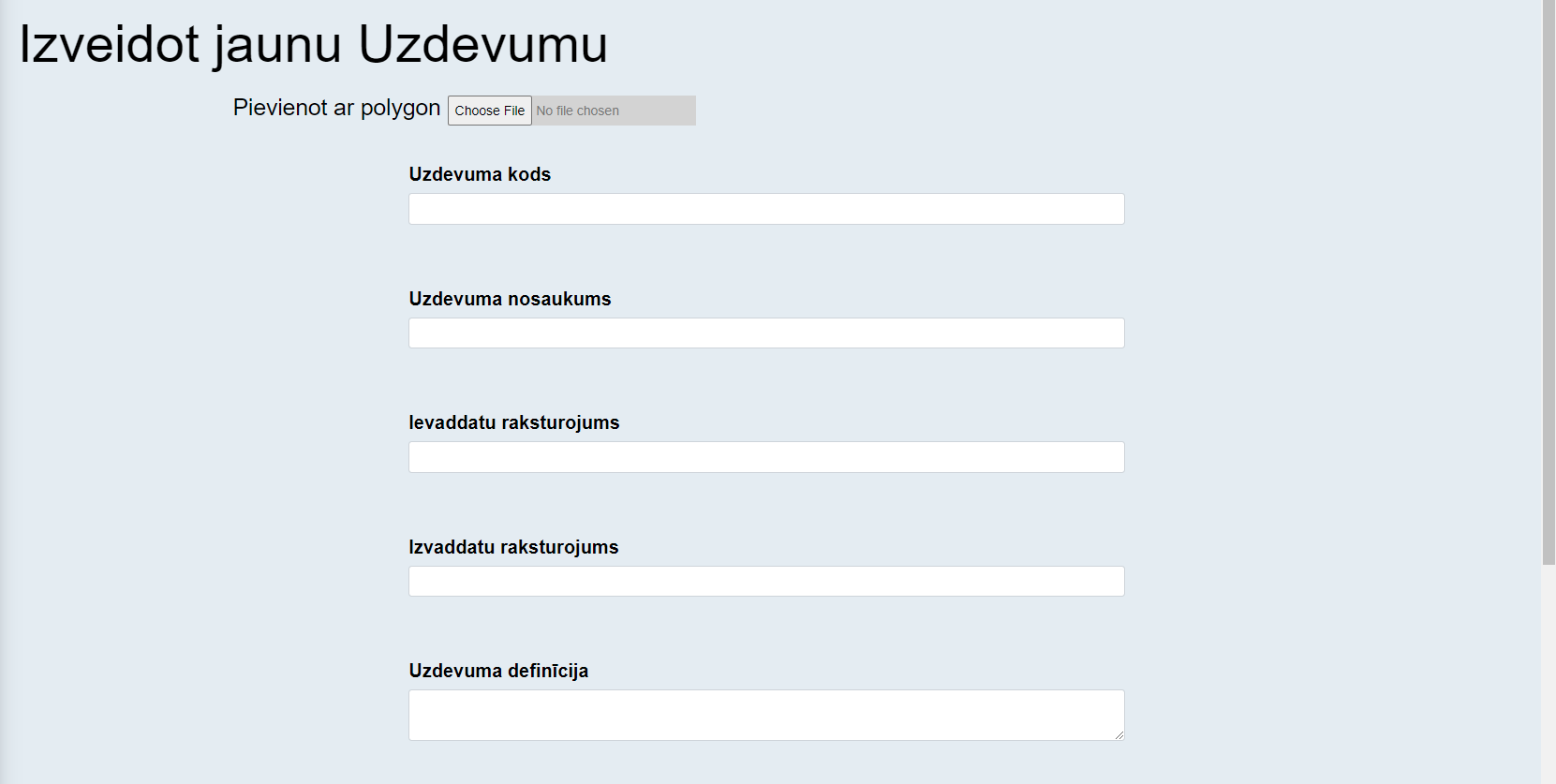
Katram iesūtījumam ir iespējams individuāli apskatīt iesūtījuma kodu un testu rezultātus, nospiežot uz iesūtījuma sarakstā. Nospiežot uz iesūtījuma ieraksta, tiek atvērts modulis ar visiem publiski redzamajiem iesūtījuma datiem.



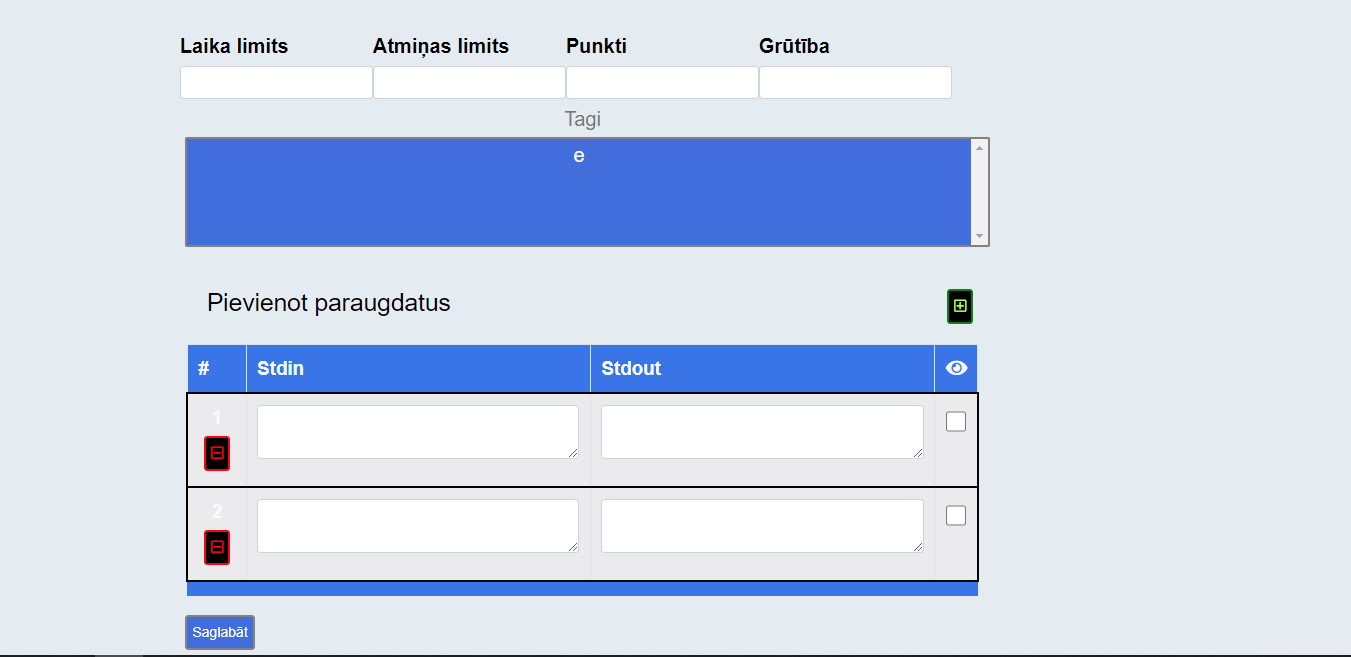
6.10. att. Konkrēta iesūtījuma atvērts modulis, ar uzdevuma testu rezultātiem un kodu

**Uzdevuma izveidošana**

Lai izveidotu uzdevumu nepieciešams aizpildīt uzdevuma izveidošanas formu, balstoties uz datu plūsmas kritērijiem. Uzdevuma labošanas skats ir identisks uzdevuma izveidošanas skatam, vienīgā atšķirība ir tā, ka anketa tiek aizpildīta ar uzdevumu vecajām vērtībām.



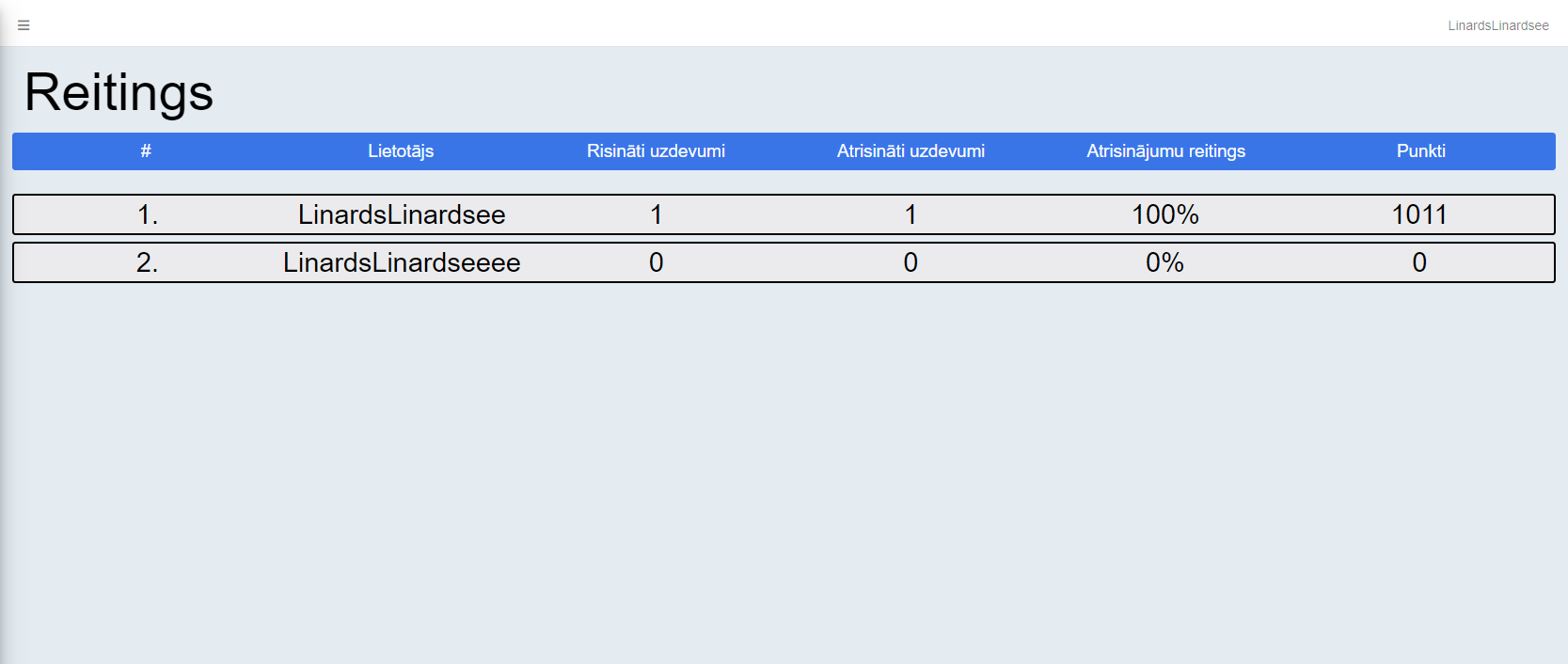
6.11. Uzdevuma izveidošanas anketas pirmā puse



6.12. Uzdevuma izveidošnas anketas otrā puse

**Reitinga attēlošana**

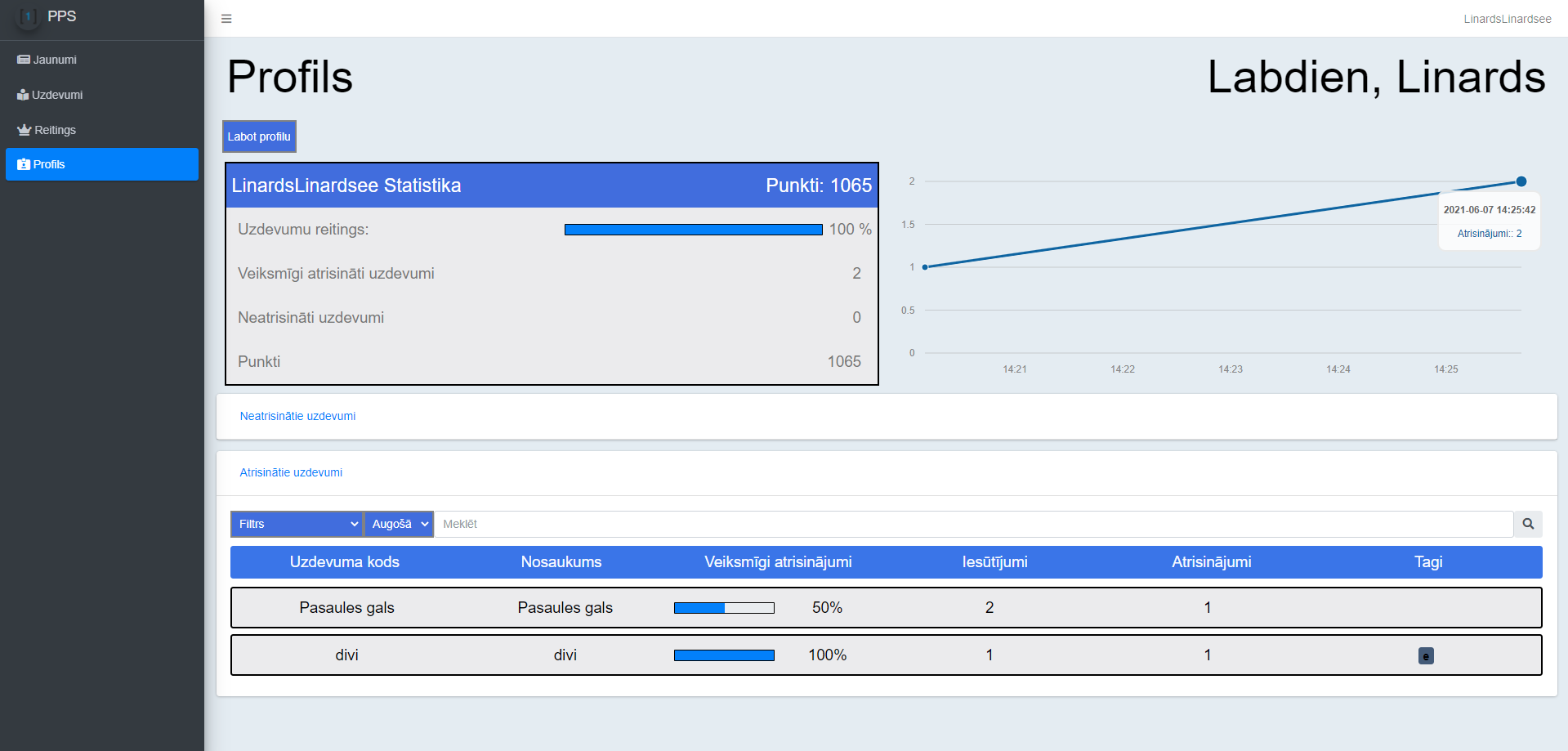
Lai attēlotu kopējo lietotāju reitingu, tiek izmantots reitinga skats, kur visi lietotāji tiek sakārtoti pēc punktiem un uzrādīti saraksta formā. Nospiežot uz jebkuru no lietotājiem, tiek atvērts lietotāja profils



6.13. att. Reitinga skata attēlošana

**Profila informācijas attēlošana**

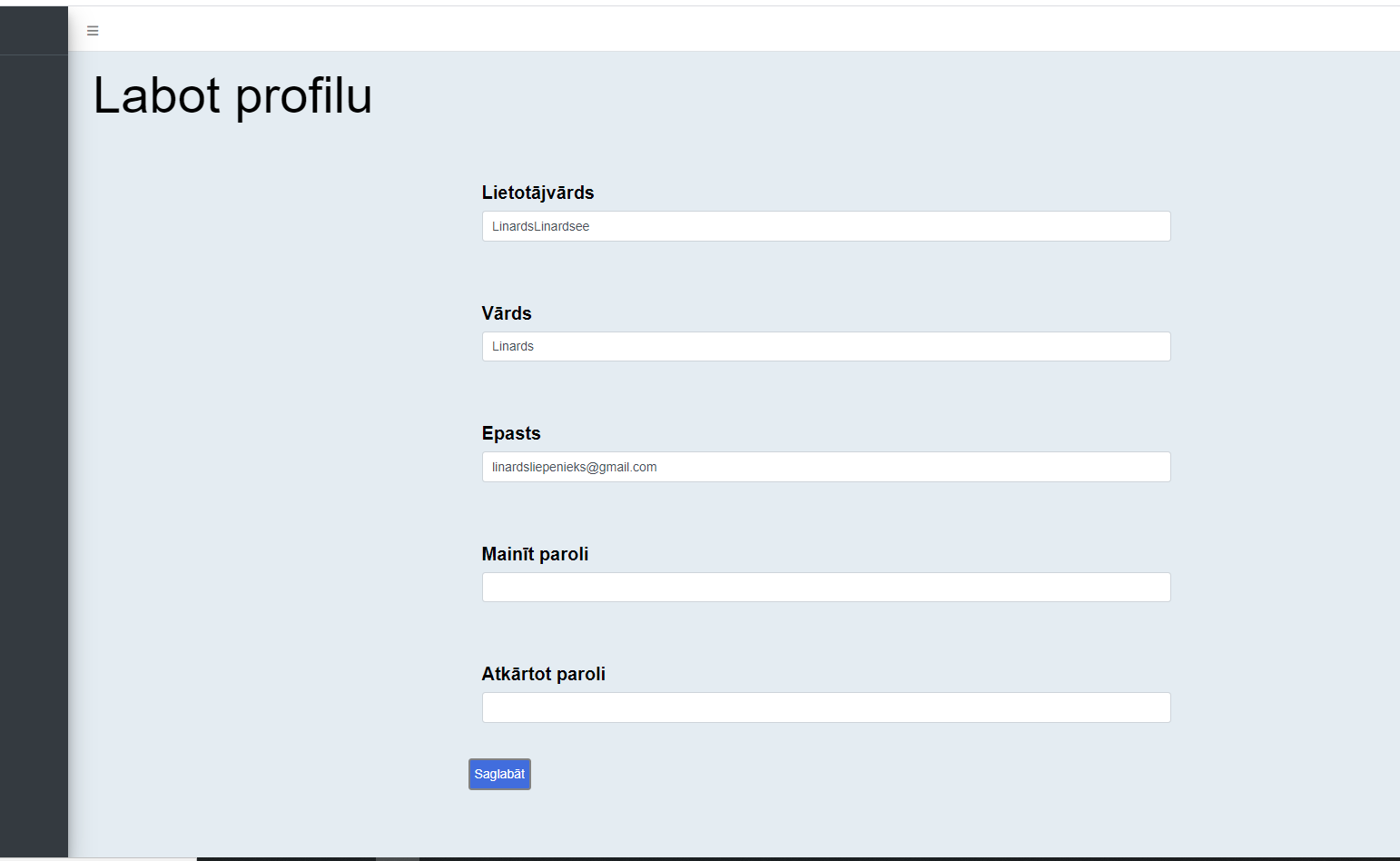
Jebkuram lietotājam ir iespēja apskatīt savu, vai citu lietotāju profilus. Ja lietotājs apskata savu profilu, tad ir iespējams labot savus profila datus nospiežot uz pogas “Labot profilu”.



6.14. att. Sava profila attēlošanas skats

**Profila labošnaas skats**

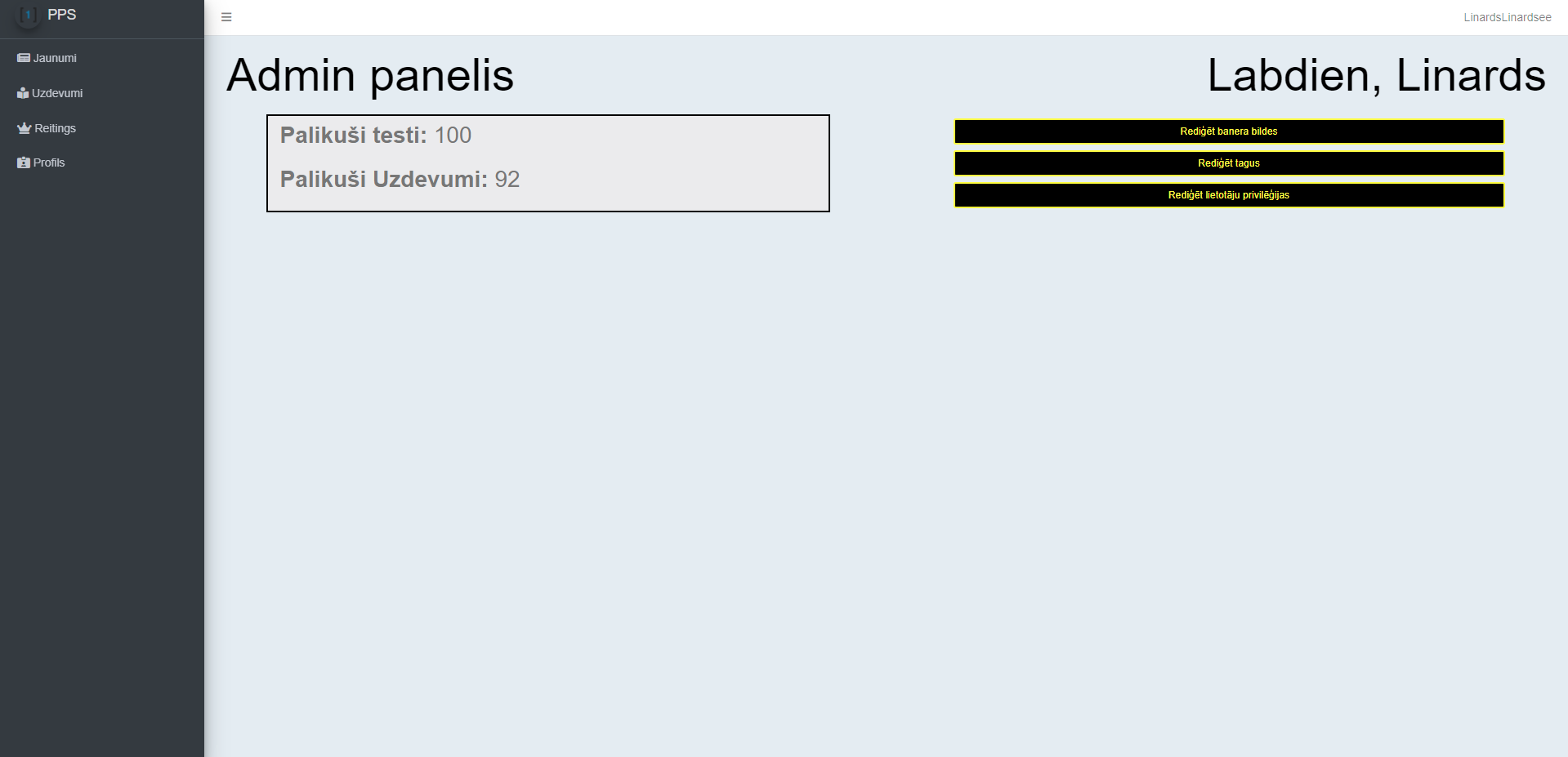
Lietotājam ir iespēja, pēc reģistrācijas rediģēt sava profila informāciju: lietotājvārdu, vārdu, epastu un paroli, ja tiek ievēroti datu plūsmas kritēriji. Atverot skatu Lietotājvārda, vārda un epasta informācija, tiek aizpildīta ar lietotāja pašreizējo informāciju.



6.15. att. Profila labošanas skats

**Admin panelis**

Ja lietotājs ir admins, caur lietotāja izvēlni, ir iespējams nonākt admin panelī, kur iespējams rediģēt parastiem lietotājiem nepieejamu papildinformāciju un redzēt, cik Judge0 API izmantojamie testi ir palikuši – testēšanas laikā iesūtījumiem hosts tika piemērojis limitu (dienā 100 uzdevumi), tāpēc attēlā redzams šodienas palikušo, iespējamo, iesūtījumu skaits. (Skat. Att. 6.16.)



6.16. att. Admin paneļa skats

**Banera bilžu rediģēšana**

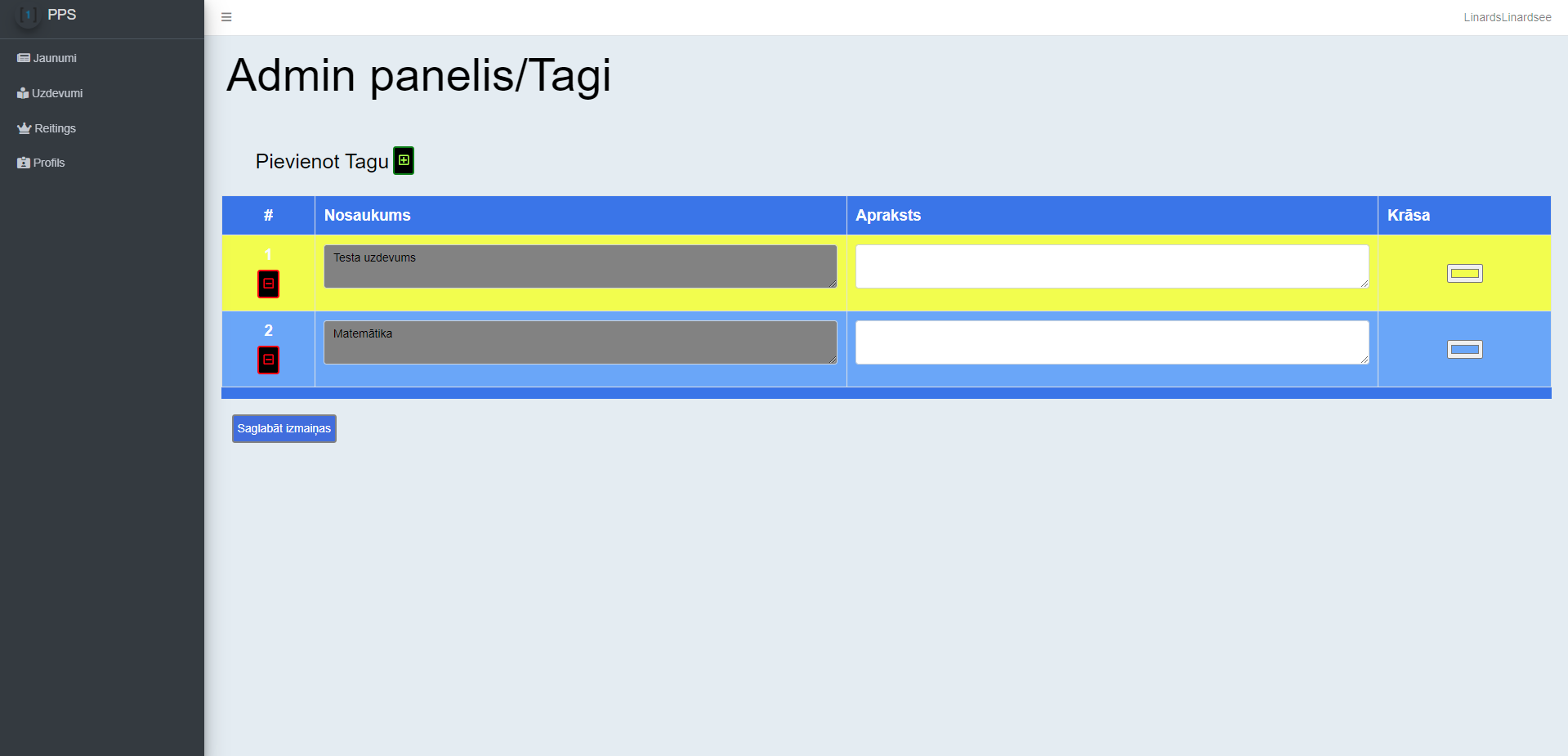
Lietotājam iespējams mainīt banerī redzamās bildes dinamiski tās pievienojot un dzēšot, svarīgi pieminēt, ka izmaiņas tiks saglabātas tikai tad, kad lietotājs nospiedīs pogu “saglabāt izmaiņas”.



6.17. att Banera bilžu redaktors

**Tagu rediģēšana**

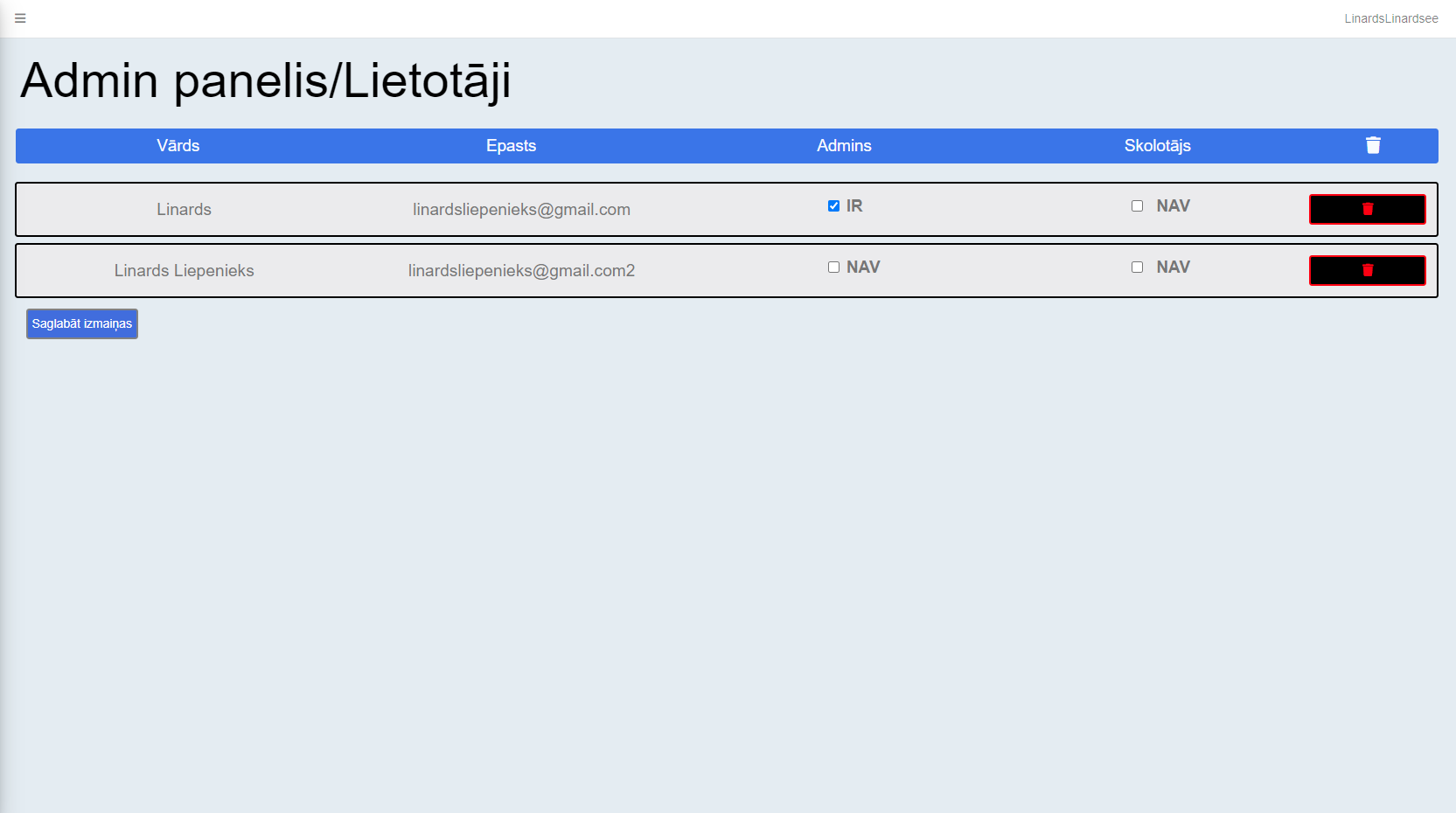
Ir iespējams rediģēt, pievienot un dinamiski dzēst tagus. Tagu skatā tiek uzrādīti visi izveidotie tagi un to rindas iekrāsotas krāsā, kādā tagi uzrādīsies. Tagu izmaiņas tiks saglabātas tikai tad, kad lietotājs nospiedīs pogu “Saglabāt izmaiņas”



6.18. att. Tagu redaktors ar diviem pievienotiem tagiem

**Lietotāju rediģēšana**

Admin lietotājiem ir iespējams rediģēt citu lietotāju privilēģijas, kā arī dzēst lietotājus no sistemas, šo ir iespējams darīt lietotāju redaktorā. Izmaiņas tiks saglabātas tikai tad, kad lietotājs nospiedīs pogu “Saglabāt izmaiņas”



6.19. att. Lietotāju redaktora skats

# NOBEIGUMS

Projekta galvenais uzdevums bija izveidot sistēmu ar kuru iespējams iesūtīt programmēšans uzdevumus un sistēma izvērtēs, vai uzdevums izpildīts pareizi. Skatoties pēc šī kritērija uzdevums ir izpildīts veiksmīgi. Nākotnē plānots implementēt ērtāku administrāciju un papildus funkcionalitātes skolotāja statusa profiliem – pašlaik šis statuss kalpo tikai kā identifikācija.

Sistēmu ir plānots izmantot no nākošā mācību gada, vasaras periodā ir plānots sākt atklātu testēšanu uzņēmuma ietvaros un secināt, vai darbība paātrina un atvieglo skolotāja darbu un padara mācību procesu elastīgāku, jo šis nozīmē, ka uzdevumus ir iespējams iesūtīt un apskatīt ārpus nodarbību laika, bez skolotāja uzraudzības – tas protams patērē mazāk līdzekļu, jo skolotājam nav jāmaksā par darba stundām. Pagaidām, aizvērtajā testēšanas periodā, atsauksmes ir bijušas pozitīvas. Pagaidām galvenie secinājumi un atsauces no testētājiem ir pozitīvi, visievērojamākie plusi, ko dod sistēma ir tas, ka ir iespējams trenēties un apskatīt uzdevumus ārpus nodarbību laika, kā arī visi risinājumi un iesūtījumi ir ērti pārskatāmi no datubāzes – kods nevar pazust un vienmēr var apskatīt citus risinājumu veidus. Skolēniem, kas vēlās pamēģināt citu programmēšanas valodu ir ērti iesūtīt jau izpildītu programmēšanas uzdevumu, bet citā valodā.

Liela problēma izstrādājot sistēmu, bija atrast ērtu veidu, kā kompilētu uzdevumus – Judge0 ir ļoti ērts risinājums. Nākotnē ir plānots attīstīt lietotāju pārvaldes un lietotāju iesūtījumu apskates sistēmas, kas nebūs problēma, jo galvenā funkcionalitāte ir izveidota.

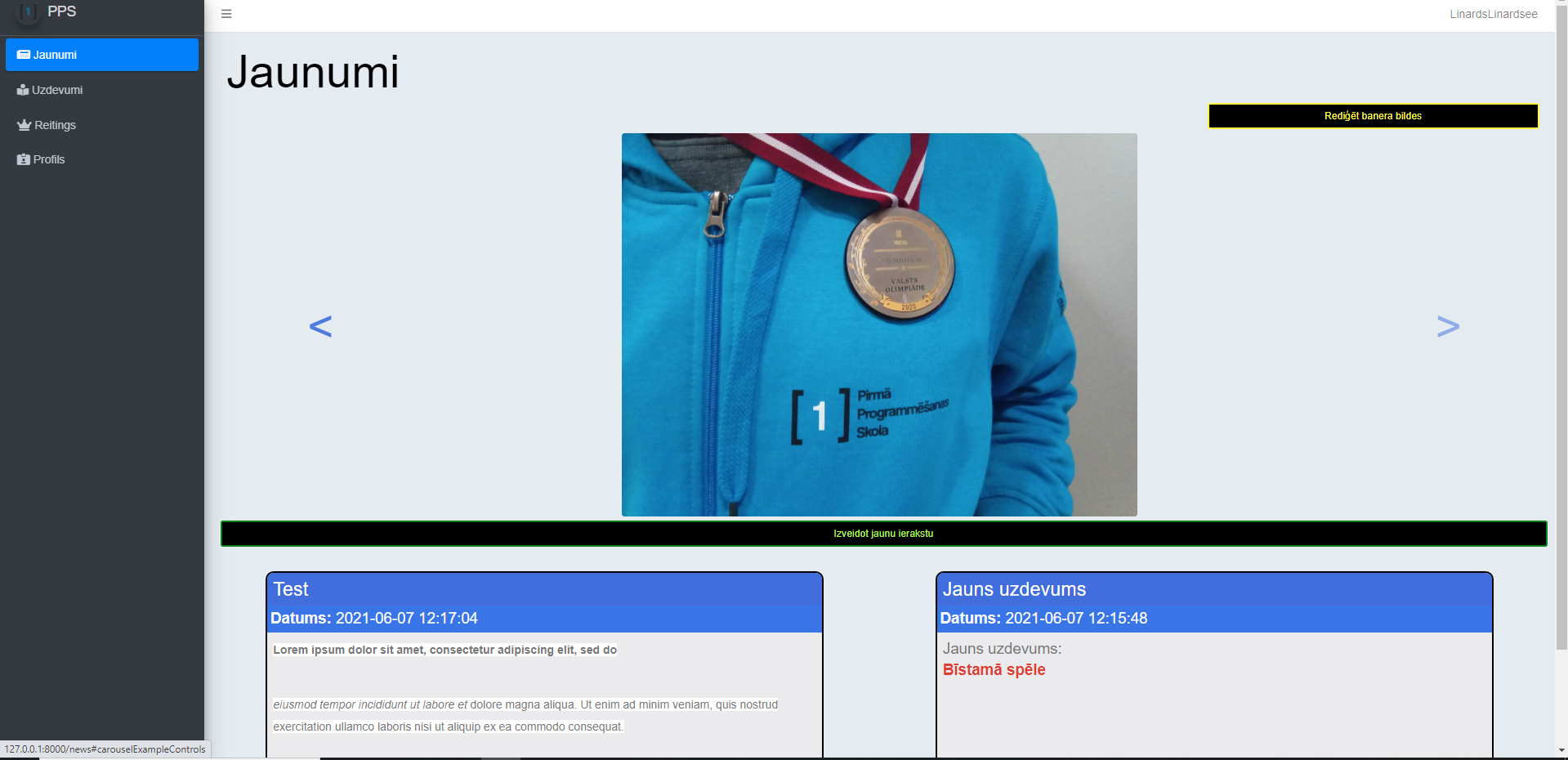
# INFORMĀCIJAS AVOTI

* Laravel ietvara dokumentācija - <https://laravel.com/docs/8.x>
* VUE.js dokumentācija - <https://vuejs.org/v2/guide/>
* CodeMirror koda redaktors - <https://codemirror.net/>
* TinyMCE teksta redaktors - <https://www.tiny.cloud/>
* Composer dependeny manager - <https://getcomposer.org/>
* NodeJS - <https://nodejs.org/en/docs/>
* MorrisJS - <https://morrisjs.github.io/morris.js/>
* Judge0 - <https://judge0.com/>
* Bootstrap ietvars - <https://getbootstrap.com/>
* AdminLTE veidne - <https://adminlte.io/>

**PIELIKUMI**

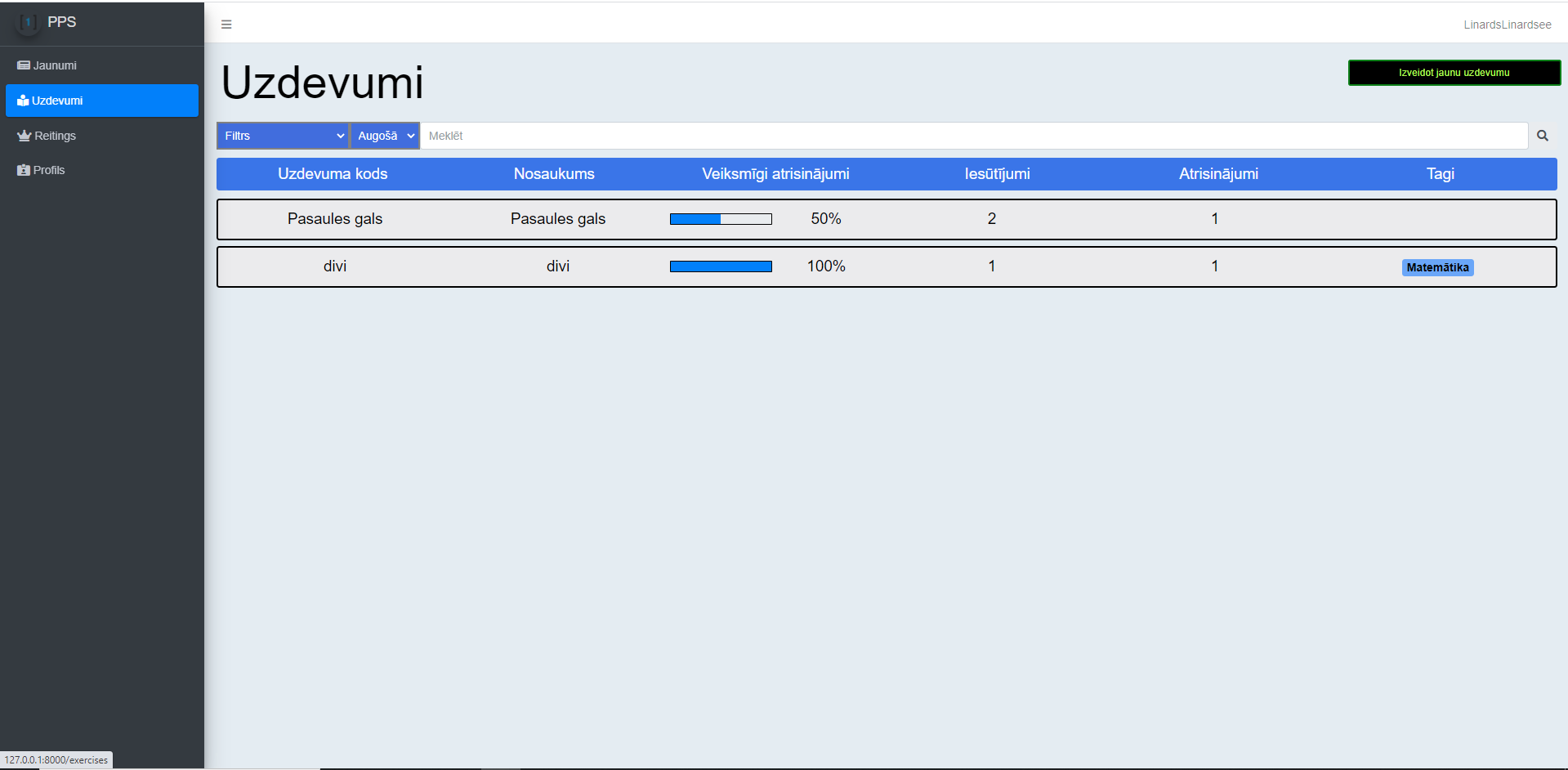
## 1. Ekrānformu piemēri

**Jaunumu skata izskats datorā (skat 1, pielikuma 1. att).**



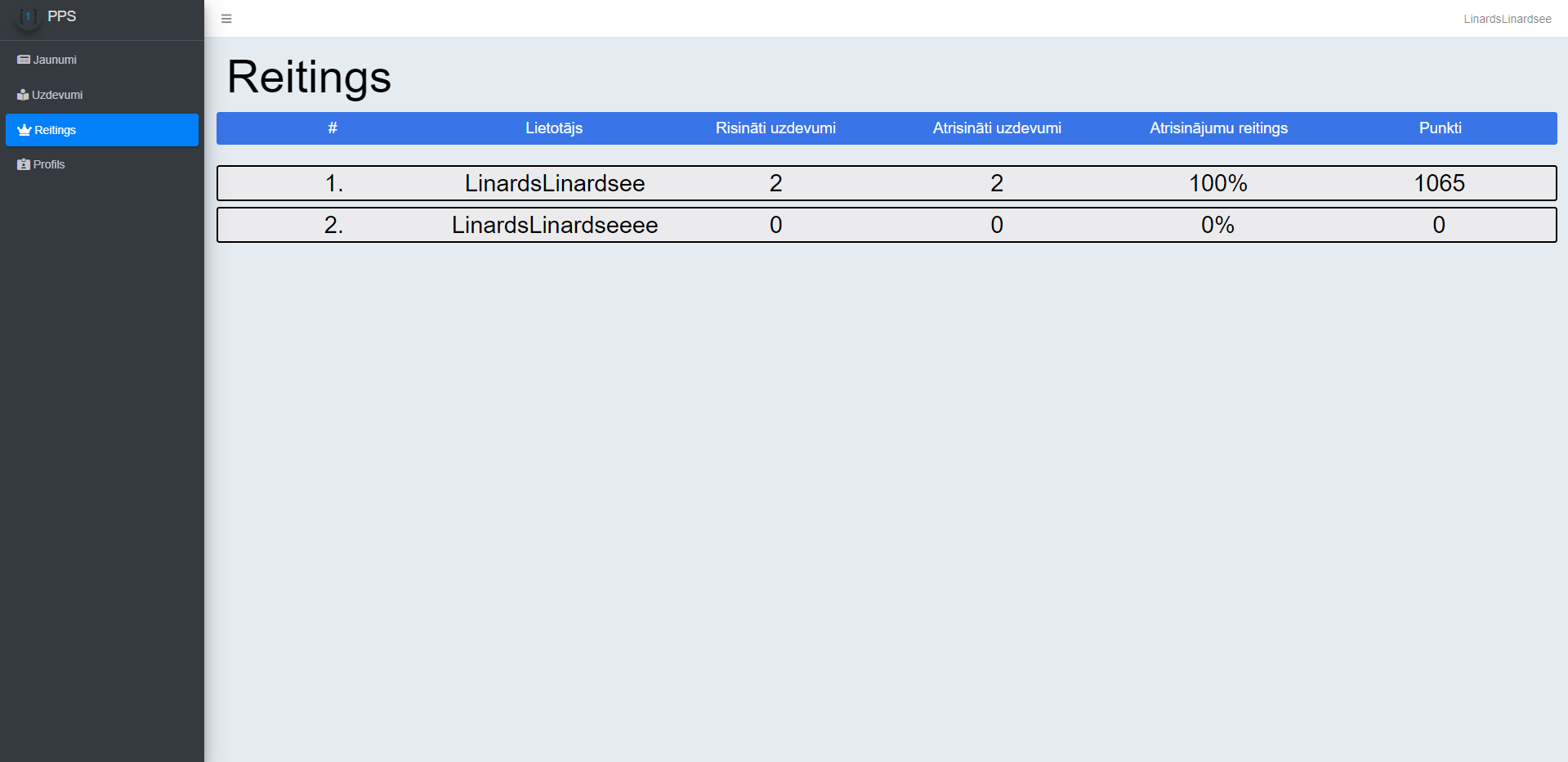
1. pielikuma 1. att Jaunumu skats datorā

**Uzdevumu skata izskats datorā (skat 1. pielikuma 2. att).**



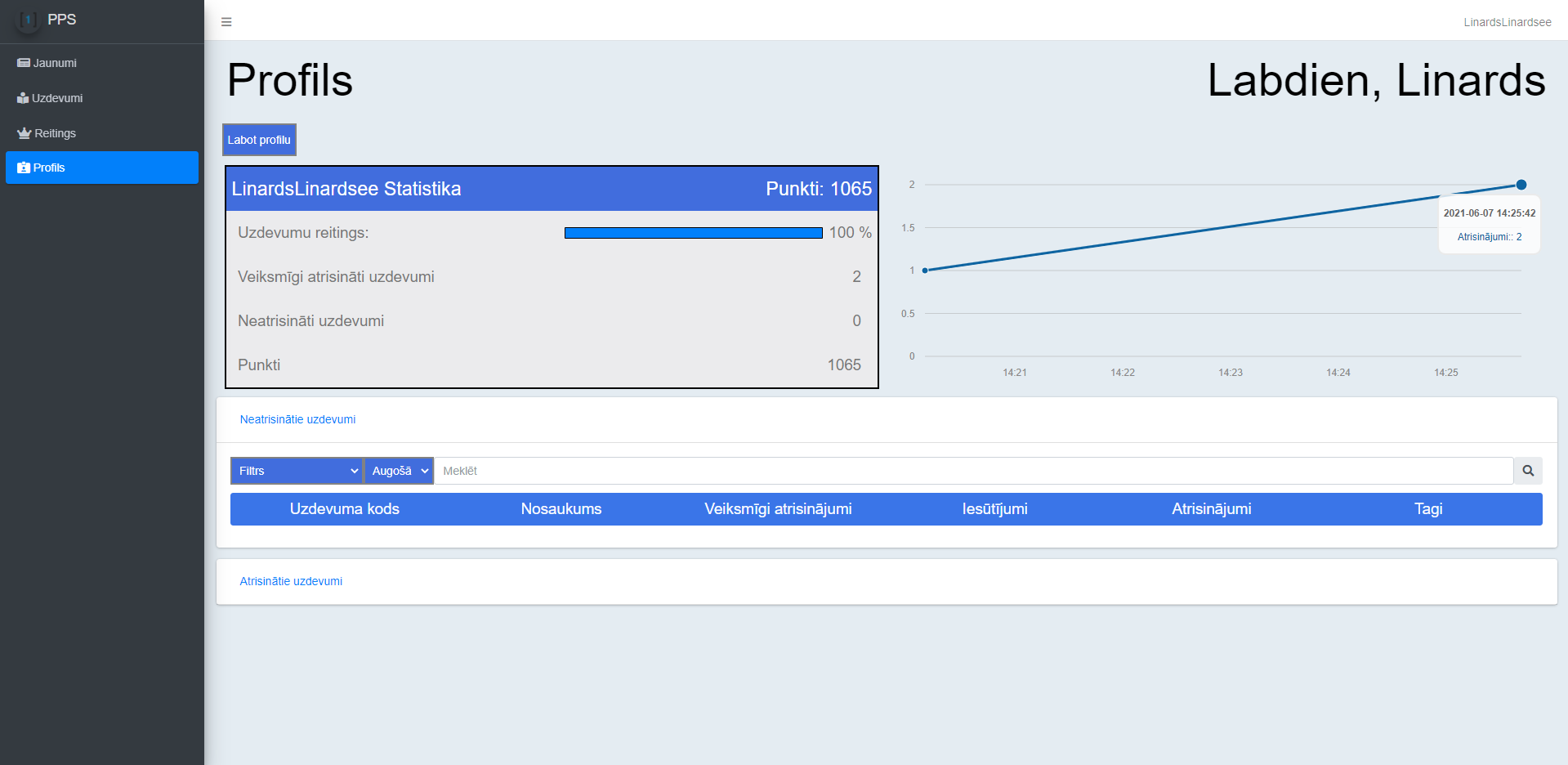
1. pielikuma 2. att Uzdevumu skats datorā

**Reitinga skata izskats datorā (skat 1. pielikuma 3. att).**



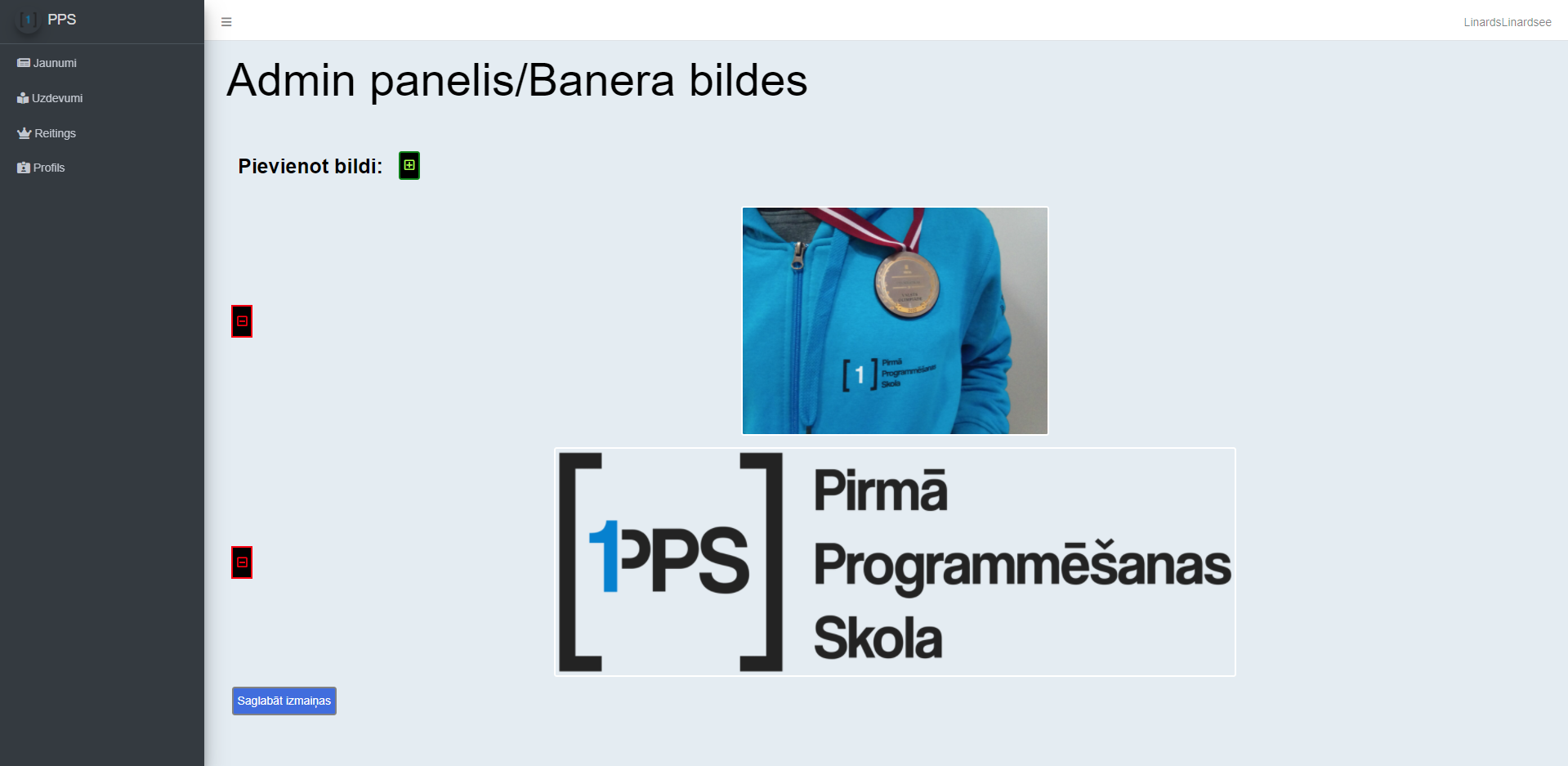
1. pielikuma 3. att Reitinga skats datorā

**Profila skata izskats datorā (skat 1. pielikuma 4. att).**



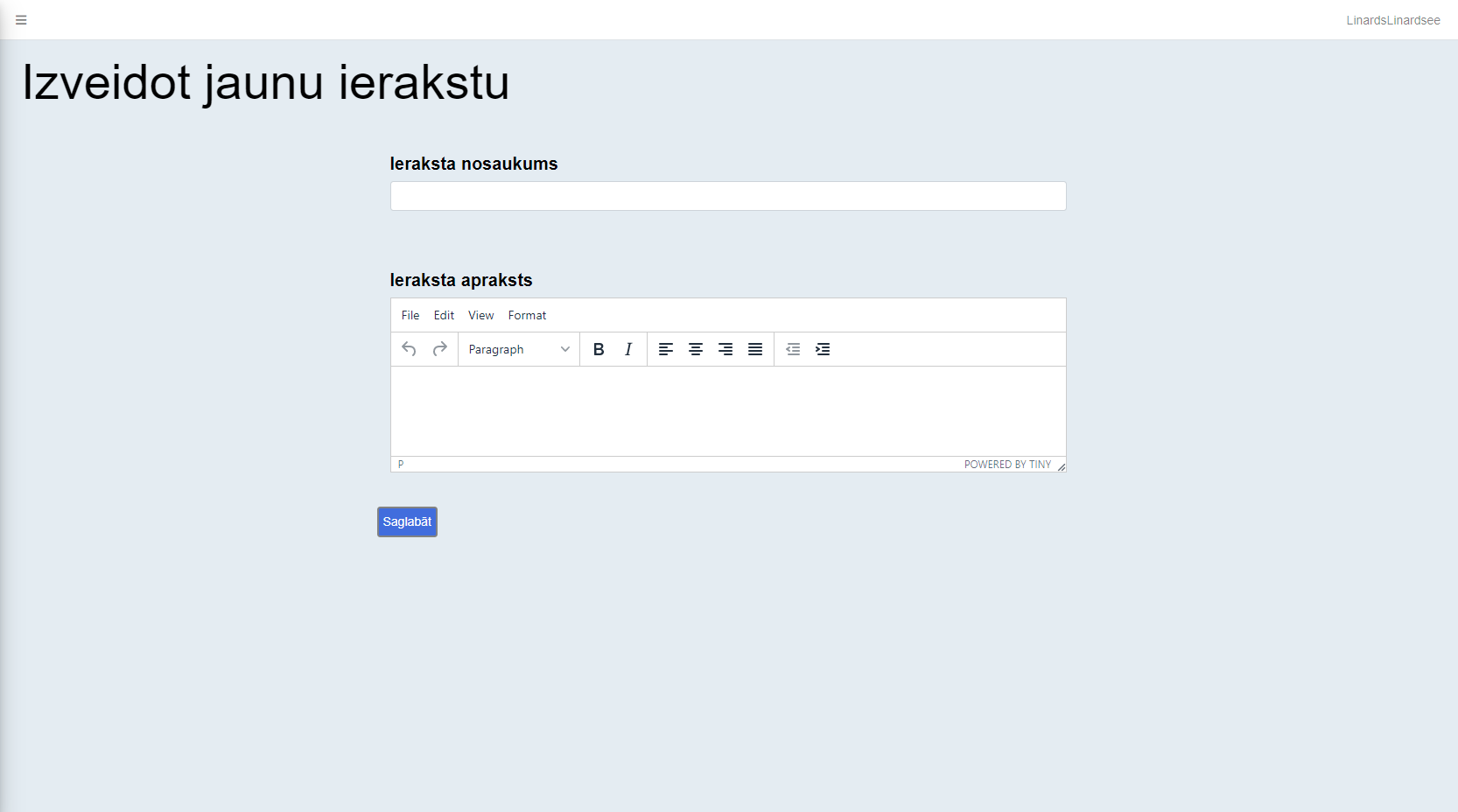
1. pielikuma 4. att Profila skats datorā

**Banera rediģēšanas izskats datorā (skat 1. pielikuma 5. att).**



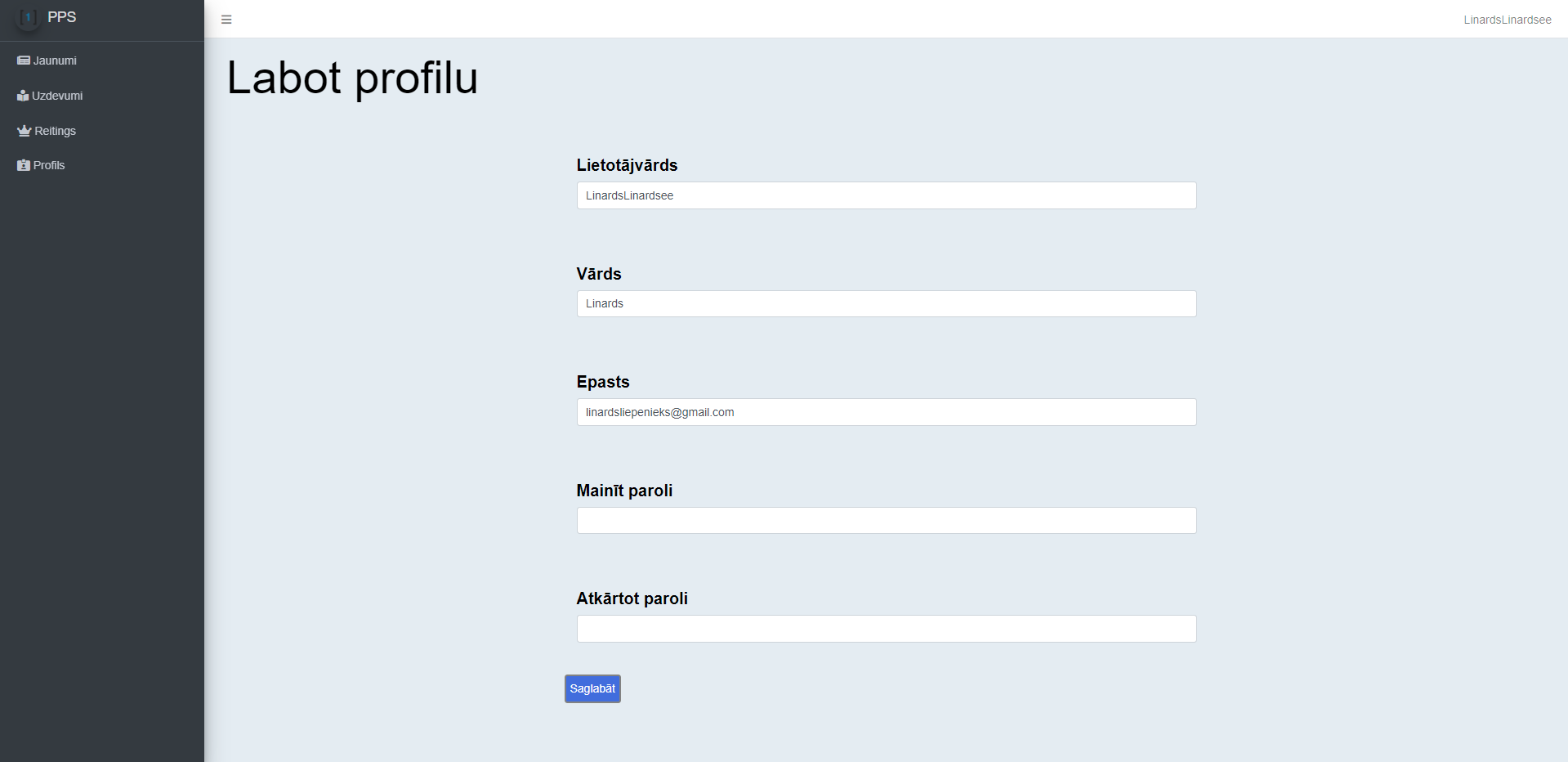
1. pielikuma 6. att Banera redaktora skats datorā

**Jaunumu ieraksta izveides loga izskats daorā (skat 1. pielikuma 7. att).**



1. pielikuma 7. att Jaunumu ieraksta izveides skats datorā

**Profila labošanas loga izskats datorā (skat 1. pielikuma 8. att).**



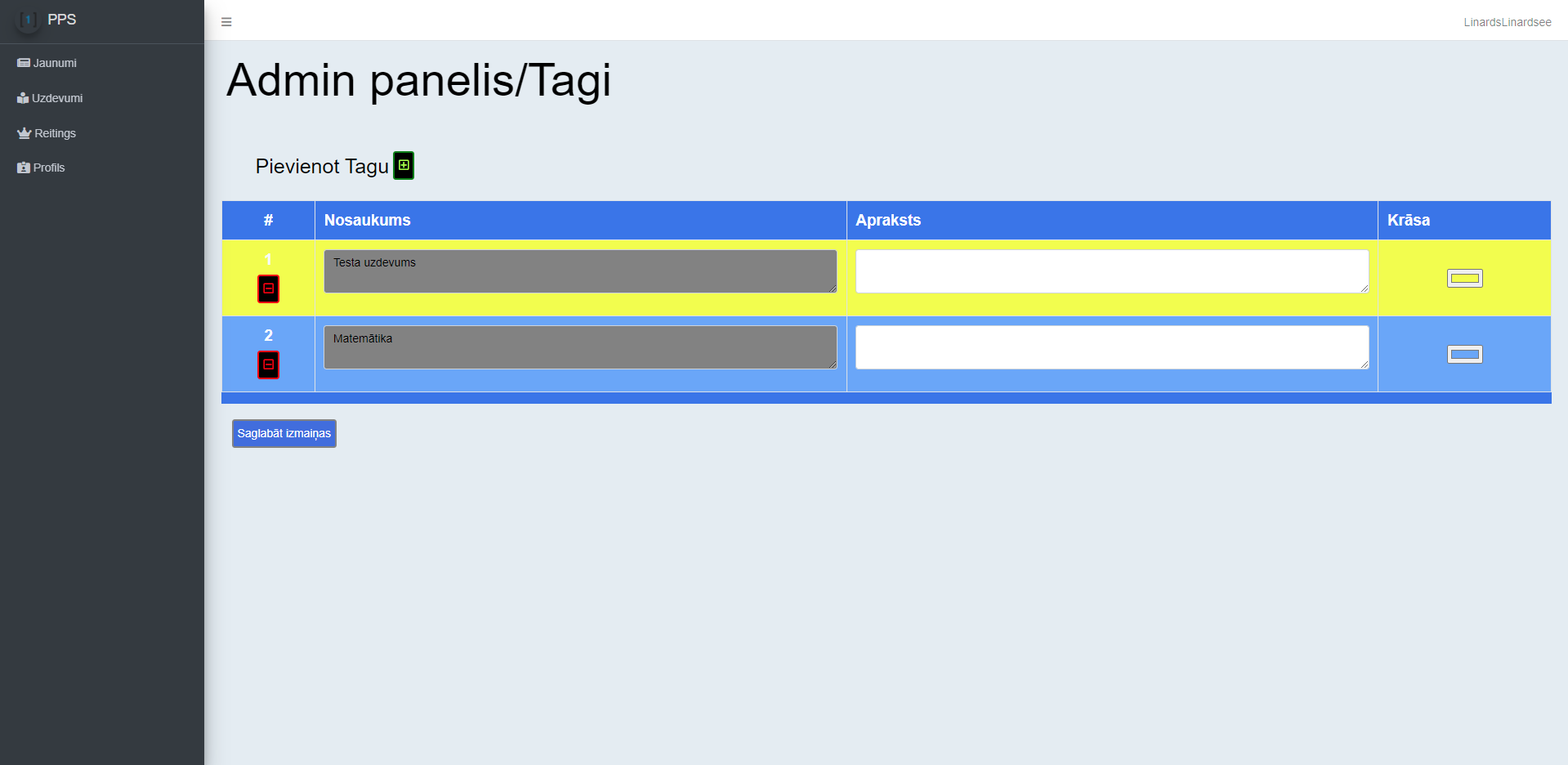
1. pielikuma 8. att Profila labošanas skats datorā

**Admin paneļa loga izskats datorā (skat 1. pielikuma 9. att).**



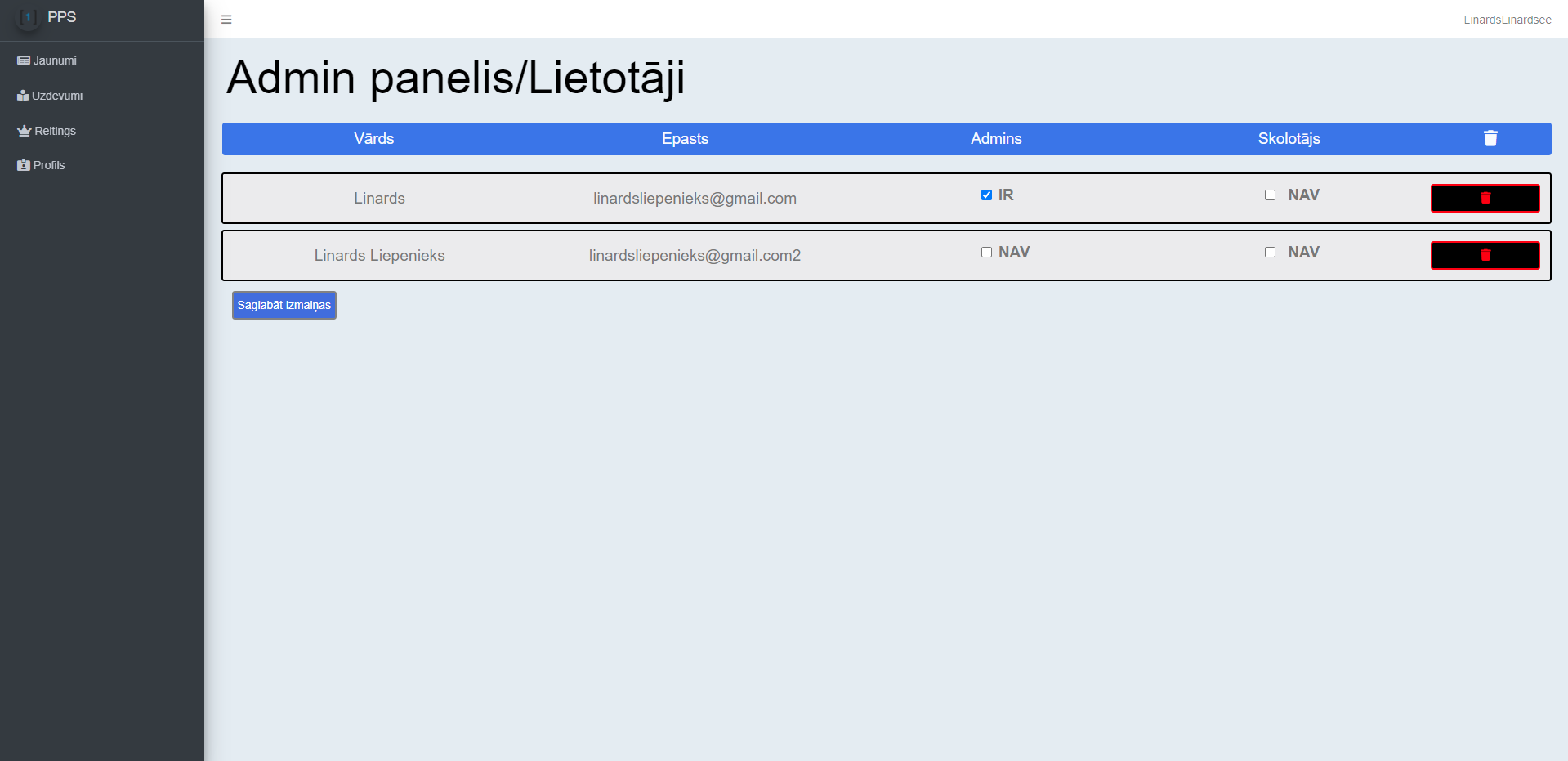
1. pielikuma 9. att Admin paneļa skata izskats datorā

**Tagu redaktora loga izskats datorā (skat 1. pielikuma 10. att.)**



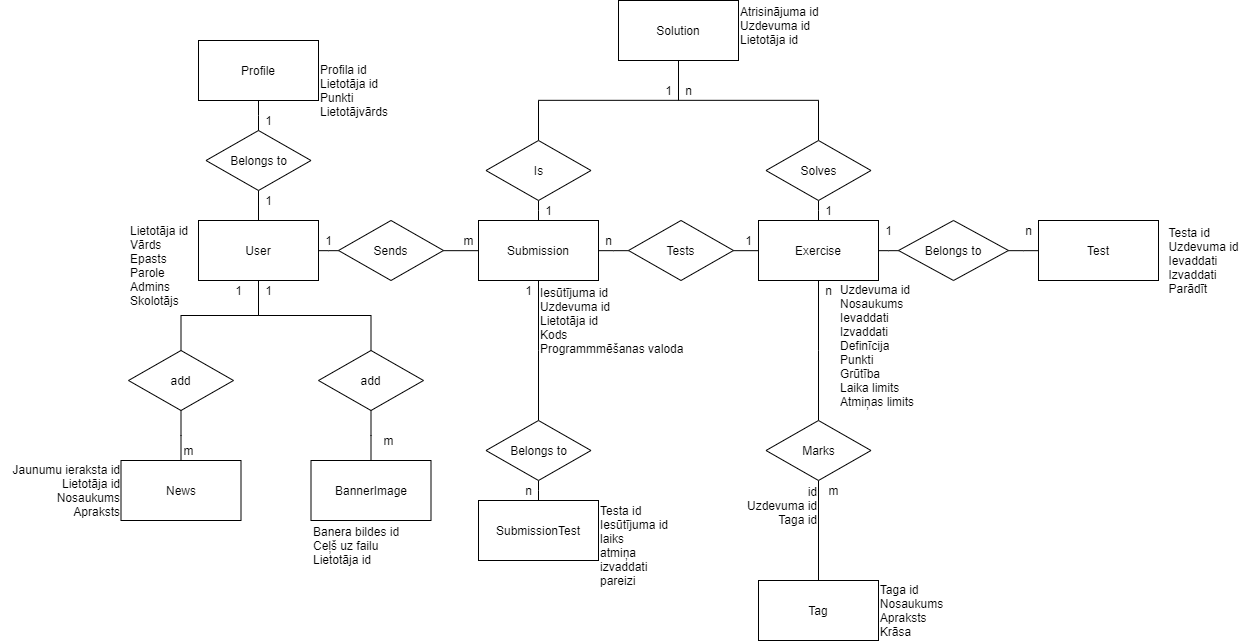
1. pielikuma 10. att Tagu redaktora izskats datorā

**Lietotāju redaktora loga izskats datorā (skat 1. pielikuma 11. att)**

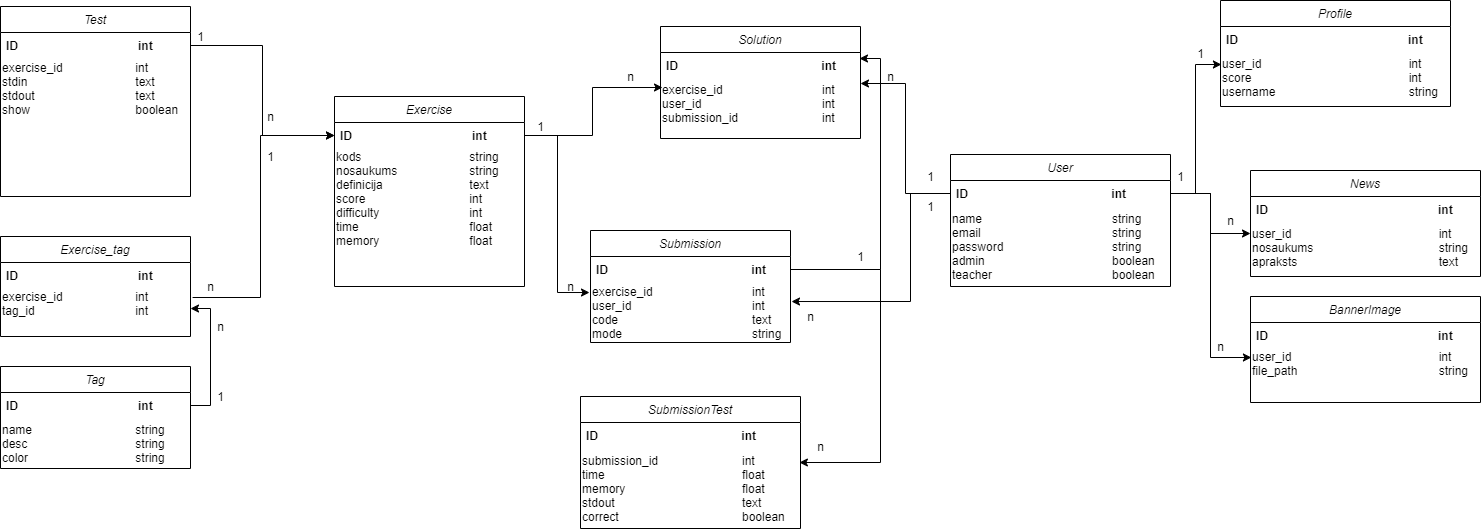


1. pielikuma 11. att Lietotāju redaktora skats datorā

## 2. pielikums ER diagramma



## 3. pielikums tabulu saišu shēma



## 4. pielikums Sistēmas arhitektūra

