VENTSPILS AUGSTSKOLA
INFORMĀCIJAS TEHNOLOĢIJU FAKULTĀTE

BAKALAURA DARBS

Docētāju noslodzes pārvaldības sistēmas izveide.

Autors

Ventspils Augstskolas
Informācijas tehnoloģiju fakultātes
bakalaura studiju programmas „Datorzinātnes”
3. kursa students
Endijs Bertāns
Matr.nr. 22020008

_____
(paraksts)

Fakultātes dekāns doc. Dr.sc.comp. Vairis Caune

_____
(paraksts)

Zinātniskais vadītājs _____
(ieņemamais amats, zinātniskais nosaukums, vārds, uzvārds)

_____
(paraksts)

ANOTĀCIJA

Darba nosaukums: Docētāju noslodzes pārvaldības sistēmas izveide.

Darba autors: Endijs Bertāns

Darba vadītājs: doc. Dr.sc.comp. Vairis Caune

Darba apjoms: * lpp., * tabulas, * attēli, * bibliogrāfiskie norādes, * pielikumi.

Atslēgas vārdi: SPRING BOOT, SPRING SECURITY, JWT, ANGULAR, AWS.

Bakalaura darba mērķis ir izstrādāt docētāju slodzes pārvaldības sistēmu pēc Informācijas tehnoloģiju fakultātes dekāna prasībām.

Sistēmas izstrādei izmantots Java Spring Boot ietvars kopā ar Spring Data JPA, Spring Security un Spring Boot Starter Validation bibliotēkām, kā arī datu bāzes sistēma MySQL. Lietotāja saskarne izstrādāta, izmantojot Angular. Sistēmas darbība palaista Docker konteinerizētā vidē, tādejādi sistēmu var palaist uz jebkuras operētājsistēmas, kurā atrodas docker.

Darba gaitā izstrādāta tīmekļa lietotne, kas ļauj plānot un pārvaldīt docētāju slodzes.

ABSTRACT

Title: Development of a workload management system for teaching staff.

Author: Endijs Bertāns

Academic Advisor: doc. Dr.sc.comp. Vairis Caune

Volume of the work: * pages, * tables, * images, * equations, * literature

Keywords: SPRING BOOT, SPRING SECURITY, JWT, ANGULAR, AWS.

The aim of the bachelor thesis is to develop a workload management system for teaching staff according to the requirements of the Dean of the Faculty of Information Technology.

The system is developed using the Java Spring Boot framework with Spring Data JPA, Spring Security and Spring Boot Starter Validation libraries, as well as the MySQL database system. The user interface was developed using Angular. The system runs in a Docker containerised environment.

A web application has been developed to allow the scheduling and management of lecturers' workloads.

Saturs

IEVADS

Mūsdienās valsts un privātām iestādēm arvien būtiskāka kļūst efektīva, viegla un droša procesu pārvaldība, lai šos procesus atvieglotu un paātrinātu. Ventspils Augstskolas semestra plānu apstiprināšanas process notiek relatīvi manuāli – semestra plānus veido *excel* izklājlapā, kur atrodas tabula ar 41 kolonu, katru mācību gadu veidojot jaunu izklājlapu, kas var novest pie aizkavējumiem un lielas darbinieku slodzes. Tā rezultātā palielinās kļūdu risks.

Šīs problēmas risināšanai ir plānots izstrādāt docētāju noslodzes pārvaldības sistēmu, kuras mērķis ir automatizēt semestra plānu veidošanas procesu, nodrošināt labāku informācijas apmaiņu starp sistēmas lietotājiem, piemēram, docētājiem un dekāniem, kā arī nodrošināt datu saglabāšanu vienā sistēmā, lai varētu viegli pārskatīt citu mācību gadu datus.

Lai sasniegtu izvirzīto mērķi, pētījumā tiks veikti sekojoši uzdevumi:

- Apskatīt esošo slodzes pārvaldības sistēmu.
- Apskatīt dažādas datu bāzes, lai secinātu kuru šai sistēmai izmantot būs vispiemērotāk.
- Apskatīt dažādas aizmugursistēmas.
- Apskatīt pieejamos drošības risinājumus izvēlētajām sistēmām.
- Apskatīt dažādus risinājumus, sistēmas saskarnes izveidei.
- Modelēt un izstrādāt datubāzi, pēc esošo excel izklājlapu, piemēriem.
- Realizēt projektēto datubāzi.
- Manuāli ievadīt paraugdatus no izklājlapām, lai pārliecinātos, ka tiek saglabāta esošā funkcionalitāte un visiem datiem no izklājlapas ir vieta datubāzē.
- Izveidot aizmugursistēmu saskarnei ar datubāzes tabulām - datu izveidei, lasīšanai, rediģēšanai un dzēšanai.
- Ieviest sistēmā drošības risinājumus.
- Izmantojot atlasīto saskarnes ietvaru, veidot pieslēgšanos sistēmai, izmantojot izvēlēto drošības risinājumu
- Izveidot datu izvades / ievades saskarni.
- Izveidot sistēmai automātisku palaišanu uz serveri, pēc jaunāko izmaņu nosūtīšanu versiju kontroles rīkam.

Darba struktūra ir loģiski sadalīta vairākās daļās, sākot ar teorētisko daļu, kurā apskatīti semestra plānu izveides procesa esošie risinājumi un to ierobežojumi, apskatīti ietvari un tehnoloģijas sistēmas veidošanai, turpmāk detalizēti izklāstīta jaunās sistēmas projektēšana un izstrāde, un beidzoties ar praktisko sistēmas izveidi. Pētījuma temats ir norobežots ar konkrētām Ventspils augstskolas vajadzībām, taču metodes un pieejas var tikt paplašinātas arī citās iestādēs ar līdzīgi strukturētām administratīvajām prasībām.

Šajā darbā izmantotās pētījuma realizācijas metodes ietver esošās sistēmas procesu analīzi, jaunu risinājumu projektēšanu un prototipa izstrādi, tādejādi tiks nodrošināta administratīvā procesa un automatizācija Ventspils Augstskolā.

1. ESOŠĀS SISTĒMAS ANALĪZE

Lai izprastu nepieciešamās prasības un funkcionalitāti jaunajai centralizētai sistēmai nepieciešams analizēt esošo sistēmu, lai nepazaudētu funkcionalitāti, risinātu esošās problēmas un automatizētu iespējami vairāk funkciju, tādejādi uzlabojot un paātrinot semestra plānu izveides procesu.

Pašreizējā procesa ietvaros datu ievade notiek *excel* izklājlapā ar 41 kolonu, kur katrā rindā ir ieraksts specifiskam priekšmetam, vai tā priekšmeta teorijas daļai un, vai praktiskai daļai, kur tiek ievadīta sekojoša informācija:

- docenta vārds,
- docenta uzvārds,
- vārds un uzvārds kopā,
- amats un vārds, un uzvārds,
- kredītpunkti pilnai slodzei, kuri iegūti no docenta amatu grupas, piemēram, lektoriem kredītpunkti pilnai slodzei ir 12
- Slodzes daļa, kura nosaka formula (esošās rindas kredītpunkti * grupu skaitu * koeficentu) / kredītpunktiem pilnai slodzei
- Docenta amata nosaukums, piemēram, ja profesors ir ievēlēts:
    - profesors
    - asoc. Profesors
    - docents
    - asistents
    - lektors

  bet, ja profesors nav ievēlēts

    - viesprofesors
    - viesasoc. Profesors
    - viesdocents
    - viesasistents
    - vieslektors
- Amata grupa, piemēram,
    - profesors
    - asoc. Profesors
    - docents
    - asistents
    - lektors
- Statuss, piemēram,
    - Neievēlētie
    - Ievēlētie
- Iekļaut budžetā, piemēram,
    - 1
    - Nenotiks
    - 0
- Pasniedzēja fakultāte, piemēram,
    - ITF
    - TSF
- Semestris, piemēram,
    - Rudens,
    - Pavasaris
- Priekšmeta nosaukums, piemēram,

    - Mehānika,
    - Sensori un robotika
- Programmas koeficents, kuru aprēķina iegūstot attiecību starp kredītpunktiem un kontaktstundām (maģistriem 0.75, bakalauriem 1)
- Priekšmeta kredītpunkti
- LAIS kods
- Reģistrācija, piemēram,
    - Automātiska
    - Izvēles
- Priekšmeta fakultāte
- Programma, piemēram,
    - KNE
    - EIB
- Grupa semestra grafikam, piemēram,
    - 1KNE
    - 1EIB
- Grupa, piemēram,
    - 1EIB+1KNE+LiepU
- Grupu skaits
- KP skaits grupai
- KpxGrupasxKoef kuru aprēķina reizinot kredītpunktus ar grupu skaitum un ar koeficentu
- Komentāri, kur bieži vien norādīts vai ieraksts domāts praktiskām vai teorijas lekcijām.
- Akad. h./ nedēļā, kurš netiek aizpildīts
- Programmas daļa, piemēram,
    - Nozares profesionālās specializācijas kursi
    - Nozares (profesionālās darbības jomas) teorētiskie pamatkursi un informācijas tehnoloģiju kursi
    - Izvēles daļas kursi

Tālāk seko budžeta plānošanas kolonas

- Alga, kuru ņem no docenta amatu grupas.
- Nozares koeficents – palielina vieslektoru algu līdz 30%
- Alga mēnesī
- Vai atvaļinājums ieskaitās – uz vieslektoriem neattiecas
- Mēnešu skaits – atvaļinājuma mēneši + nostrādātie mēneši
- Algai paredzēts – mēnešu skaits * alga mēnesī
- Budžeta pozīcija, kura nosaka budžetu no kura lektoram maksās algu.

Katra mācību gada sākumā šīs kolonas tiek manuāli aizpildītas, izmantojot iepriekšējā gada datus kā paraugu, kas prasa ievērojamu laika ieguldījumu un nav viegli pārredzams.

Šajā tabulā atrodas dati par visiem docentiem un priekšmetiem attiecīgajā mācību gadā. Tā kā dati ir daudz un tie tiek ievadīti manuāli, pastāv augsta kļūdu ievades iespēja, kas var novest pie grūti pamanāmām kļūdām. Turklāt šajā tabulā atrodas informācija par visiem docentiem, tādejādi izklājlapu publicēšana ir ierobežota, un dot iespēju docentiem pārskatīt savu individuālo mācību gada plānu ir pārāk laikietilpīgi, jo pārskats katram

docentam būtu jātaisa atsevišķi un manuāli, taču tas ļautu docentiem novērtēt plāna atbilstību savām prasībām un dotu iespēju ieteikt uzlabojumus, kas samazinātu kļūdainību risku un palielinātu procesa caurredzamību.

Datu pārskatīšanai un analīzei tiek izmantotas izklājlapu iebūvētās filtrēšanas funkcijas (piemēram, “Sākas ar”, “Beidzas ar”, “Ietilpst”, “Ir vienāds”), kas ļauj atlasīt tikai tās rindas, kurām atbilst noteikts kritērijs.

Kopsavilkumā, esošā semestra plāna izveides procesa galvenās problēmas saistītas ar manuālu datu ievadi, sliktu caurredzamību un nepietiekamu validācijas mehānismu. Jaunā sistēma ievērojami uzlabotu procesa caurredzamību, samazinātu kļūdu iespējamību un samazinātu cilvēk-resursu izmantošanu, tādējādi nodrošinot labāku administratīvo procesu izpildi.

2. UZDEVUMA NOSTĀDNE.

Bakalaura darba uzdevums ir pamatojoties uz esošās sistēmas analīzi identificēt un formulēt mērķus jaunās sistēmas izstrādei, kura būs paredzēta augstskolas administrācijas, docētāju un programmu direktoru izmantošanai.

Mērķis ir izveidot risinājumu, kas automatizē datu ievadi, nodrošina datu validāciju pirms ievades un piedāvā ievades laukiem piemērus, lai lietotājam būtu skaidrs, kādi dati ir jāievada. Pēc datu ievades sistēmai jāspēj automātiski veikt aprēķinus un aizpildīt attiecīgos laukus, tādējādi samazinot manuālo darbu un potenciālo kļūdu iespējamību.

Sistēmai jānodrošina iespēja importēt datus no izklājlapām (piemēram, Excel formātā), kas ļaus nodrošināt vienkāršu un ātru pāreju no pašreizējās sistēmas uz jauno risinājumu.

Jaunajai sistēmai jābūt pielāgotai dažādu lietotāju grupu vajadzībām:

Administratīvie darbinieki – var ievadīt, dzēst un rediģēt datus, kā arī pievienot docentus.

Docenti – tiesības pārskatīt savu plānu, ar ierobežotu informācijas daudzumu, bet bez iespējas veikt izmaiņas pašā plānā.

Svarīgs nosacījums ir spēja katra mācību gada sākumā izveidot jaunu semestri, izmantojot iepriekšējo gada semestra datus kā bāzi. Kā rezultātā iesākt jaunu mācību semestra plānu būs ātri un vienkārši.

Lietotāja saskarnei jābūt intuitīvai un vizuāli līdzīgai izklājlapai, kas ļaus lietotājiem viegli pielāgoties jaunajai sistēmai. Tāpat jānodrošina iespēja filtrēt un sakārtot datus līdzīgos principos kā esošajā sistēmā.

Jāievēro industrijas labās prakses drošības aspekti, tostarp datu drošā dzēšana – izdzēstie dati netiek pilnībā dzēsti no sistēmas, bet gan paslēpti, lai tos nepieciešamības gadījumā varētu atgriezt. Tas ļaus izvairīties no nejaušas datu dzēšanas tādējādi sabojājot esošos ierakstus.

Analizējot pašreizējo slodzes pārvaldības sistēmu, secināms, ka esošo pieeju, kuras pamatā ir manuāla datu ievade un Excel izklājlapu izmantošana, ir nepieciešams uzlabot. Galvenie uzdevumi ir:

- Nodrošināt datu pārbaudi,
- Nodrošināt automātisku aprēķināmo lauku aizpildi,
- Nodrošināt pieeju docentiem, ar iespēju apskatīt savu mācību gada plānu,
- Nodrošināt datu ievadi no izklājlapas
- Nodrošināt iepriekšējā mācību gada plānu kopēšanu uz esošo mācību gadu.

Jaunās centralizētās sistēmas izstrāde ļaus apvienot vairākas būtiskas funkcijas: automatizētu datu apstrādi, uzlabotu datu validāciju, lietotāju lomu diferenciāciju un dinamisku datu atjaunošanu. Tas veicinās caurredzamību un lietotāju iesaisti, dodot docentiem iespēju aktīvi piedalīties mācību gada plāna veidošanā.

3. PRASĪBU SPECIFIKĀCIJA

3.1 Funkcionālās prasības

1. Jānodrošina lietotāju autentifikācija un autorizācija:
   1.1. sistēmā autentificējas izmantojot epastu un paroli;
   1.2. sistēmā ir iespējams nomainīt paroli neautorizējoties;
   1.3. sistēma paziņo par nepareizi ievadītiem datiem, ja autorizācija ir bijusi neveiksmīga
   1.4. ja lietotājs iepriekš ir autorizējies, nepieciešams automātiski autorizēt viņu.
   1.5. servera autorizācijai jāizmanto JWT žetoni.
2. Jānodrošina lietotāju izveide ar neobligātu konta piesaisti:
   2.1. jāievada vārds, uzvārds, statuss (vai docētājs ir pilna laika ilgtermiņa augstskolas darbinieks), fakultāte un amats;
   2.2. ja lauki atstāti tukši neļaut saglabāt lietotāju un izvadīt kļūdas paziņojumu;
   2.3. sistēmā nepieciešams izvēlēties vai docētājam veidot kontu:
      2.3.1. administratoram reģistrējot jaunu lietotāju nepieciešams ievadīt epastu un lomu(piekļuves tiesības);

2.3.2. pēc lietotāja un konta izveides, uz lietotāja norādīto epastu sistēma nosūta kodu un apstiprināšanas saiti;

2.3.3. saite aizved lietotāju uz sistēmas lapu, kur jaievada kods, un pēc koda ievades pieprasa ievadīt savu paroli un apstiprināt paroli ievadot to velreiz.

3. Jānodrošina lietotāju ievade no csv faila;
   3.1. atrodoties lietotāju izveides skatā nepieciešama poga, kas aizved uz izveidi no csv faila
   3.2. jāatrodas pogai ar kuru tiek lejupielādēts paraugs csv failam
   3.3. csv failā jāievadā jāievadā vārds, uzvārds, statuss (vai docētājs ir pilna laika ilgtermiņa augstskolas darbinieks), fakultāte un amats;
   3.4. reģistrējot jaunu lietotāju nepieciešams ievadīt epastu un lomu(piekļuves tiesības), ja tie nav ievadīti konts netiek veidots;
   3.5. pēc lietotāja un konta izveides, uz lietotāja norādīto epastu sistēma nosūta kodu un apstiprināšanas saiti;
   3.6. saite aizved lietotāju uz sistēmas lapu, kur jāievadā kods, un pēc koda ievades pieprasa ievadīt savu paroli un apstiprināt paroli ievadot to velreiz;
   3.7. ja csv failā atrodas duplikāti vai kļūdaini ieraksti, tos nesaglabāt datubāzē un izvadīt kļūdas paziņojumu;
4. sistēmā jānodrošina saskarni ar datubāzi:
   4.1. izveidot jaunus ierakstus;
   4.2. rediģēt ierakstus
   4.3. ierakstus nedzēst, bet gan atzīmēt tos kā dzēstus.
5. sistēmas galvenā tabula:
   5.1. kolonu nosaukumi ņemti no esošās sistēmas;
   5.2. jānodrošina atsevišķu kolonu filtrācija;
   5.3. datus vienlaicīgi rāda par vienu semestri;
   5.4. jānodrošina rediģēšana;
   5.5. jānodrošina jaunu ierakstu veidošana;
   5.6. jānodrošina dzēšana;
   5.7. jānodrošina izvēlēties cik ierakstus rāda vienlaicīgi;
   5.8. jānodrošina lappušu navigācija;
   5.9. sistēmā iespējams mainīt kolonu iestatījumus:
      5.9.1. iestatījumi nodrošina iespēju paslēpt kolonas;
      5.9.2. iestatījumi saglabājas datubāzē;

5.9.3. iestatījumi ir piesaistīti lietotāja profilam un tie ir privāti – tos redz tikai lietotājs;

5.9.4. iestatījumus var dzēst, rediģēt un izveidot;

6. Sistēmas galvenā tabulā ir sekojošas kolonas:
   6.1. amats, vārds, uzvārds
   6.2. vārds
   6.3. uzvārds
   6.4. KP pilnai slodzei
   6.5. slodzes daļa
   6.6. amata nosaukums
   6.7. amata grupa
   6.8. statuss
   6.9. iekļaut budžetā
   6.10. pasniedzēja fakultāte
   6.11. priekšmeta nosaukums
   6.12. programmas koeficents
   6.13. semestris
   6.14. programmas daļa
   6.15. reģistrācija
   6.16. lais kods
   6.17. KP skaits grupai
   6.18. KPxGrupasxKoef
   6.19. KP/stundas
   6.20. grupu skaits
   6.21. kontaktstundas
   6.22. kursa līmenis
   6.23. programma
   6.24. kurss
   6.25. komentāri
   6.26. alga
   6.27. budžeta pozīcija
   6.28. nozares koeficents
   6.29. alga mēnesī
   6.30. vai atvaļinājums ieskaitās
   6.31. mēneši kopā

6.32. algai paredzēts

7. sistēmā jābūt pārskata lapai:
   7.1. datus rāda par specifisku semestri
   7.2. opcija mainīt semestri.
   7.3. redzamas kopējās kontaktstundas;
   7.4. redzami kopējie kredītpunkti;
   7.5. redzama kopējā alga;
   7.6. redzama alga mēnesī;
   7.7. grafiski attēlots izmantojot pīrāga diagramu kredītpunktu sadalījums pa kursiem;
   7.8. redzama tabula, kura paskaidro grafiskā attēlojuma redzamos datus
      7.8.1. kolona ar nosaukumu kursi;
      7.8.2. kolona ar nosaukumu kredītpunkti – norāda cik kredītpunkti attiecīgajam kursam;
   7.9. grafiski attēlots izmantojot pīrāga diagramu kredītpunktu sadalījums pa studiju priekšmetiem
      7.9.1. kolona ar nosaukumu studiju priekšmets;
      7.9.2. kolona ar nosaukumu kredītpunkti – norāda cik kredītpunkti attiecīgajam studiju priekšmetam;
   7.10. grafiski attēlots izmantojot stabu diagramu kredītpunktu sadalījums pa fakultātēm;
      7.10.1. kolona ar nosaukumu fakultāte;
      7.10.2. kolona ar nosaukumu kredītpunkti – norāda cik kredītpunkti attiecīgajai fakultātei;

3.2 Nefunkcionālās prasības

1. Sistēmas saskarnes valodai ir jābūt latviešu valodā.
2. Sistēmai ir jābūt atdalītai lietotāju saskarnei un aizmugursistēmai:
   2.1. Nepieciešams izveidot aizmugursistēmu, kura nodrošinās saskarni ar datubāzes datiem.
   2.2. Nepieciešams izveidot lietotāju saskarni, kura sazinās ar aizmugursistēmu, lai iegūtu datus.
3. Sistēma strādā uz populārākajām pārlūkprogrammām, kā, “Chrome”, “Edge”, “FireFox”, “Safari”.
4. Paroles datubāzē saglabātas šifrētā formā.
5. Dati tiek saglabāti relāciju datubāzē

4. TEHNOLOĢIJU SALĪDZINĀJUMS UN IZVĒLE SISTĒMAS IZSTRĀDEI

Izstrādājamai sistēmai ir nepieciešama aizmugursistēma, lietotāju saskarne, un datubāze.

Lai pamatotu datubāzes izvēli izstrādājamajai sistēmai, tika apskatīta Muhammada Saeeda publikācija [6], kurā veikts salīdzinošs pētījums par SQL Server, Oracle un MySQL datubāzu veiktspēju. Pētījumā, izmantojot datubāzi ar 324 500 ierakstiem un veicot septiņus dažādus eksperimentus, autors ir izdarījis divus nozīmīgus secinājumus par MySQL priekšrocībām:

Pirmkārt, Saeed norāda uz MySQL atvērtā koda licences sniegtajām priekšrocībām uzņēmumiem, uzsverot, ka "MS SQL Server and Oracle database are available at a high price while MYSQL is open source, which is the big advantage for companies" [6, 9. lpp]. Šis aspekts var būtiski ietekmēt sistēmas izmaksas un ieviešanas stratēģiju.

Otrkārt, eksperimentālie rezultāti liecina, ka MySQL demonstrē augstāku veiktspēju salīdzinājumā ar pārējām analizētajām datubāzēm, secinot, ka “MYSQL work faster and give best execution time.” [6, 9. lpp]. Ātrums ir kritisks faktors sistēmas veiktspējas un lietotāju pieredzes nodrošināšanā.

Lai gūtu ieskatu par šo datubāžu praktisko pielietojumu reālās pasaules scenārijos, tika aplūkots Alexandru Marius Bonteanu zinātniskais darbs [7]. Šajā pētījumā tiek salīdzināta MySQL, Oracle, SQL Server un PostgreSQL datubāžu veiktspēja, izmantojot Java Spring ietvaru CRUD (izveidošanas, lasīšanas, atjaunināšanas, dzēšanas) operācijām ar dažāda apjoma datu kopām (50 000, 100 000, 200 000 un 500 000 ierakstiem). Autors secina, ka kopumā MySQL uzrāda vislabāko veiktspēju. Lai gan MySQL var nedaudz piekāpties lasīšanas un rakstīšanas operāciju ātrumā, tā būtiski pārspēj pārējās datubāzes dzēšanas operācijās, kur PostgreSQL ir ievērojami lēnāks.

Lai pamatotu izstrādājamās sistēmas aizmugursistēmas tehnoloģisko izvēli, tika apskatīta Dominik Choma, Kinga Chwaleba un Mariusz Dzieńkowski publikācijas [8] analīze. Šajā pētījumā autori veica veiktspējas un uzticamības (nosūtīto pieprasījumu attiecība pret izpildītajiem pieprasījumiem) salīdzinājumu starp populārākajiem aizmugursistēmas ietvariem – Express.js, Django un Spring Boot. Lai nodrošinātu objektīvu salīdzinājumu, autori izstrādāja identisku aizmugursistēmas funkcionalitāti, izmantojot katru no minētajiem ietvariem, un testēja GET, POST, PUT un DELETE pieprasījumu izpildes laiku pieaugošam lietotāju skaitam – 1000, 2000, 4000, 8000 un 16000 vienlaicīgu lietotāju.

Pētījuma secinājumi sniedza vērtīgu ieskatu katra ietvara stiprajās un vājajās pusēs. Autori konstatēja, ka pie liela vienlaicīgu pieprasījumu skaita (16000 virtuāli lietotāji)

Express.js demonstrēja ievērojami zemāku kļūdaino pieprasījumu līmeni salīdzinājumā ar Spring Boot. Tomēr tika atzīmēts, ka ar Express.js izstrādātā aplikācija šādos apstākļos darbojās ievērojami lēnāk [8, 6. lpp].

Savukārt Spring Boot izcēlās kā ātrākais no salīdzinātajiem aizmugursistēmas ietvariem, demonstrējot arī ļoti labu uzticamību (nosūtīto un atbildēto pieprasījumu attiecību) pie lietotāju skaita no 1000 līdz 8000. Turpretī Django aizmugursistēma uzrādīja vislielāko neizpildīto pieprasījumu skaitu un bija arī vislēnākā.

Autori savus novērojumus pamatoja ar to, ka Java Spring ietvarā ir integrēti dažādi veiktspējas optimizācijas mehānismi: "Based on the above conclusions, it can be inferred that the research hypotheses have been verified. The superior results obtained with Spring Boot arise from the implementation of performance-enhancing mechanisms from the Spring framework." [8, 6. lpp].

Lai pamatotu izstrādājamās sistēmas lietotāja saskarnes tehnoloģisko izvēli, tika apskatīti 2 publikācijas PAWEŁ DYMORA 1 , MIROSŁAW MAZUREK 2 , MARIUSZ NYCZ [8] analizēja koda lielumu, vienkāršību un failu struktūru, turpretī Bogusz, D., Ciszewski, P., & Pańczyk, B. [7] analizēja šo ietvaru veiktspēju. Publikācijās tika analizēti 3 ietvari:

- Angular (Typescript),
- React (Javascript),
- Vue.js (Javascript),

PAWEŁ DYMORA 1 , MIROSŁAW MAZUREK 2 , MARIUSZ NYCZ publikācijā [8] tika veikta padziļināta trīs identisku e-komercijas lietotņu analīze, kas tika izstrādātas, izmantojot Angular, Vue un React. Pētījuma mērķis bija salīdzināt izstrādes pieredzi, koda apjomu un struktūru katrā no tehnoloģijām. Tika konstatēts, ka Angular lietotnē bija vislielākais koda apjoms, kas daļēji saistīts ar tā strukturēto pieeju un komponentu arhitektūru. React, izmantojot Styled-Components bibliotēku, demonstrēja vislielāko stila koda apjomu, kas, lai arī palielina koda rindu skaitu, nodrošina labāku komponentu izolāciju un pārvaldību. Savukārt Vue, izmantojot SCSS, piedāvāja viskompaktāko stila kodu. [8]

Analizējot kodu, sākotnēji vislielākais apjoms tika konstatēts React lietotnē. Tomēr kopējais koda apjoms Angular projektā izrādījās vislielākais. Šī stingrā struktūra un Typescript valodas obligātā izmantošana, lai arī var palielināt sākotnējo koda apjomu, veicina labāku koda uzturēšanu un mērogojamību lielos projektos. Būtiski atzīmēt, ka Angular izceļas ar izteikti strukturētu failu sistēmu, kas nodrošina labāku projekta organizāciju salīdzinājumā ar Vue un React, kuri ir elastīgi savā failu struktūrā.[8]

Veiktajos veiktspējas testos tika konstatēts [7], ka analizētajos lietošanas scenārijos labākos rezultātus demonstrēja ar Angular izstrādātā lietotne, kas piecos no astoņiem scenārijiem uzdevumus izpildīja visātrāk. Konkrētajā pētījumā Angular ne tikai pārspēja abus konkurentus lielākajā daļā testu, bet arī izcēlās ar ievērojamu ātrumu nelielu lietotnes komponentu izmaiņu apstrādē.

Interesanti, ka pētījums iezīmēja arī būtisku faktoru, kas nav saistīts ar izvēlēto izstrādes ietvaru – tīmekļa pārlūkprogrammas izvēli. Konkrēti, tika novērots, ka dažos testu scenārijos Mozilla Firefox demonstrēja pat divreiz ātrāku veiktspēju nekā Google Chrome. Tādejādi secinot, ka gala lietotāja izvēlētā pārlūkprogramma var ietekmēt lietotnes darbību pat vairāk nekā pieņemtie lēmumi par tehnoloģijām.[7]

Ņemot vērā pieejamās zinātniskās literatūras secinājumus, ir veikta stratēģiski pamatota tehnoloģiju izvēle izstrādājamajai sistēmai, kas ietver MySQL datubāzi, Java Spring aizmugursistēmai, un lietotāja saskarnei Angular.

5. JAUNĀS SISTĒMAS IZSTRĀDE

Lai izstrādātu jauno sistēmu, vispirms tika izveidots datubāzes modelis. Pēc tā izveides datubāzē ievietošu testa datus un izstrādāšu skriptu, kas nolasīs un attēlos šos datus līdzīgi kā esošajā sistēmā. Kad datubāze būs veiksmīgi izveidota un paraugdati ievietoti, tiks turpināts ar aizmugursistēmas datu slāņa modeļa izstrādi un vienkāršu kontrolieru izveidi, lai iespējami ātri varētu ieviest pamata drošības risinājumu. Pēc drošības funkcionalitātes ieviešanas savienošu lietotāja saskarni ar aizmugursistēmu, lai varētu pieslēgties sistēmai. Kad lietotāja autentifikācija būs veiksmīgi ieviesta, sistēmas attīstību, tiks veikta vienlaikus izstrādājot gan aizmugursistēmas, gan lietotāja saskarnes komponentes. Kā pirmo funkcionalitāti izstrādāšu tabulu, kas atgādina esošās sistēmas risinājumu. Pēc tam tiks pievienota iespēju izveidot jaunus objektus. Nākamajā solī tiks uzlabots datu attēlojumu, ieviešot filtrēšanas iespējas, kolonnu paslēpšanas iestatījumi un kolonu kārtošanu. Visbeidzot tiks izstrādāta funkcionalitāte jauna semestra izveidei, kas ļaus ērti pārnest datus no iepriekšējiem semestriem.

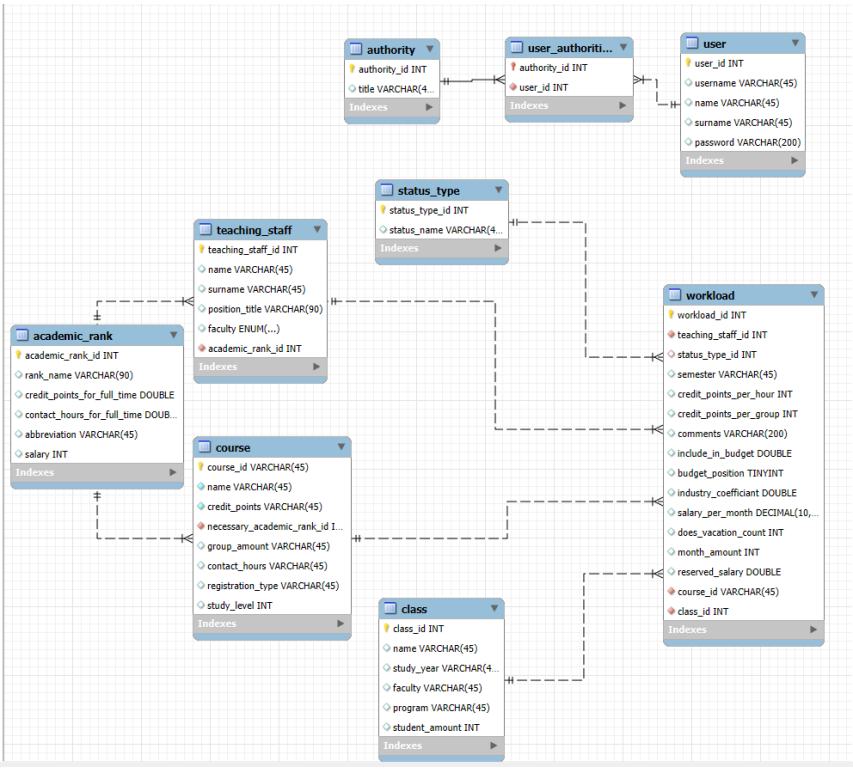
5.1 Datu bāzes modeļa izstrāde

Datubāzes pirmais modelis sastāv no 6 tabulām (4.1 att):

- Noslodzes plāns (Workload) – kura sastāv no 16 kolonām un ir domāta, kā galvenā tabula, kura atspoguļu esošās sistēmas izklajlapu, šajā tabulā atrodas atslēgas no studiju priekšmetu tabulas (course), statusa tipu

(status_type) tabulas, docētāju (teaching staff) tabulas un kursa (class) tabula, visas tabulas savienotas ar viens pret daudziem attiecību.

- Studiju priekšmeta (course) tabula sastāv no 8 kolonām, kur aprakstīts studiju priekšmets, un vienu atslēgu uz amatu grupas (academic rank) tabulu, lai norādītu kāda ir nepieciešamā amatu grupa, lai pasniegtu šo studiju priekšmetu, tabulas savienotas ar viens pret daudziem attiecību.
- Docētāju (teaching staff) tabula sastāv no 6 kolonām, kur atrodas informācija par docētāju, kā arī viena atslēga uz amatu grupas tabulu (academic rank), lai norādītu docētāja amatu, tabulas savienotas ar viens pret daudziem attiecību.
- Statusa tipa (status type) tabula, kura sastāv no 2 kolonām un neietver atslēgas no citām tabulām. Šī tabula ir, lai norādītu pasniedzēja statusu, kā, piemēram, ievēlēts, kas nozīmē, ka pasniedzējs nav vieslektors.
- Kursa (class) tabula, kura sastāv no 6 kolonām un neietver atslēgas no citām tabulām. Šī tabula norāda informāciju par kursu – kursa gadu, kursa vārdu, programmu, studentu skaitu
- Amatu grupu (academic rank) tabula sastāv no 6 kolonām, un neietver atslēgas no citām tabulām. Šī tabula norāda informāciju par amatu grupu – amata nosaukumu, kredītpunkti pilnai slodzei, kontaktstundas pilnai slodzei, abreviatūra un alga.



4.1. att. Pirmais datu bāzes modelis

Lai nodrošinātu lietotāju autentifikāciju sistēmā izveidoju arī papildus 3 tabulas:

- Lomas (authority)
- Lietotāja (user)
- Un lietotāju lomu (user authorities) tabula lai savienotu Lomas un lietotājus ar "many to many" attiecību.

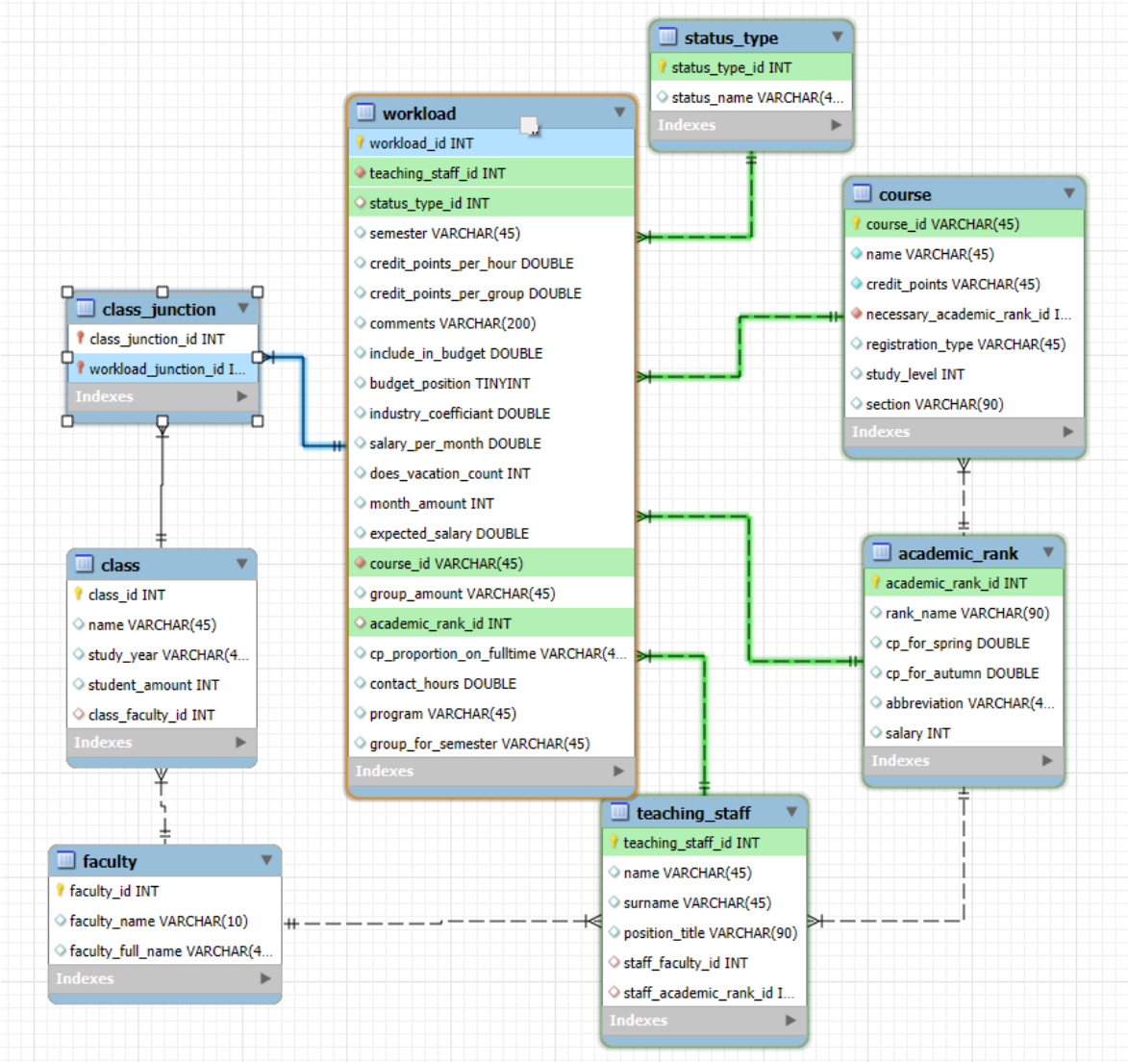
Kad modelis izveidots, datubāzei pievienoju datus no esošās sistēmas (skat. pielikumā nr.1) pēc kā jau datus liekot, atklāju sekojošas nepilnības:

- Esošais datu bāzes modelis neatbalsta pievienot noslodzes plānam vairākus kursus.
- Esošajā datu bāzes modelī mainot docētājiem amatu grupu, tā tiktu nomainīta visos ierakstos, taču amata grupa docētājam var mainīties starp mācību gadiem.
- Fakultātes kolona atkārtojas starp tabulām.

Nepilnības salabošanas rezultātā izveidojās 2 papildu tabulas (4.2 att):

- Klases savienošanas (class junction), lai spētu noslodzes plāna ierakstam pievienot vairākas klases
- Fakultāšu (faculty) tabulu, lai tabulās, kur atradās kolonas par fakultāti, tiktu izmantoti dati no šīs tabulas.

Kā arī noslodzes plāna (workload) tabulai tika pievienota kolonu ar atslēgu amatu grupai, tādejādi ļaujot docētājiem mainīt amata grupu, dažādos mācību gados, neietekmējot citu gadu datus.



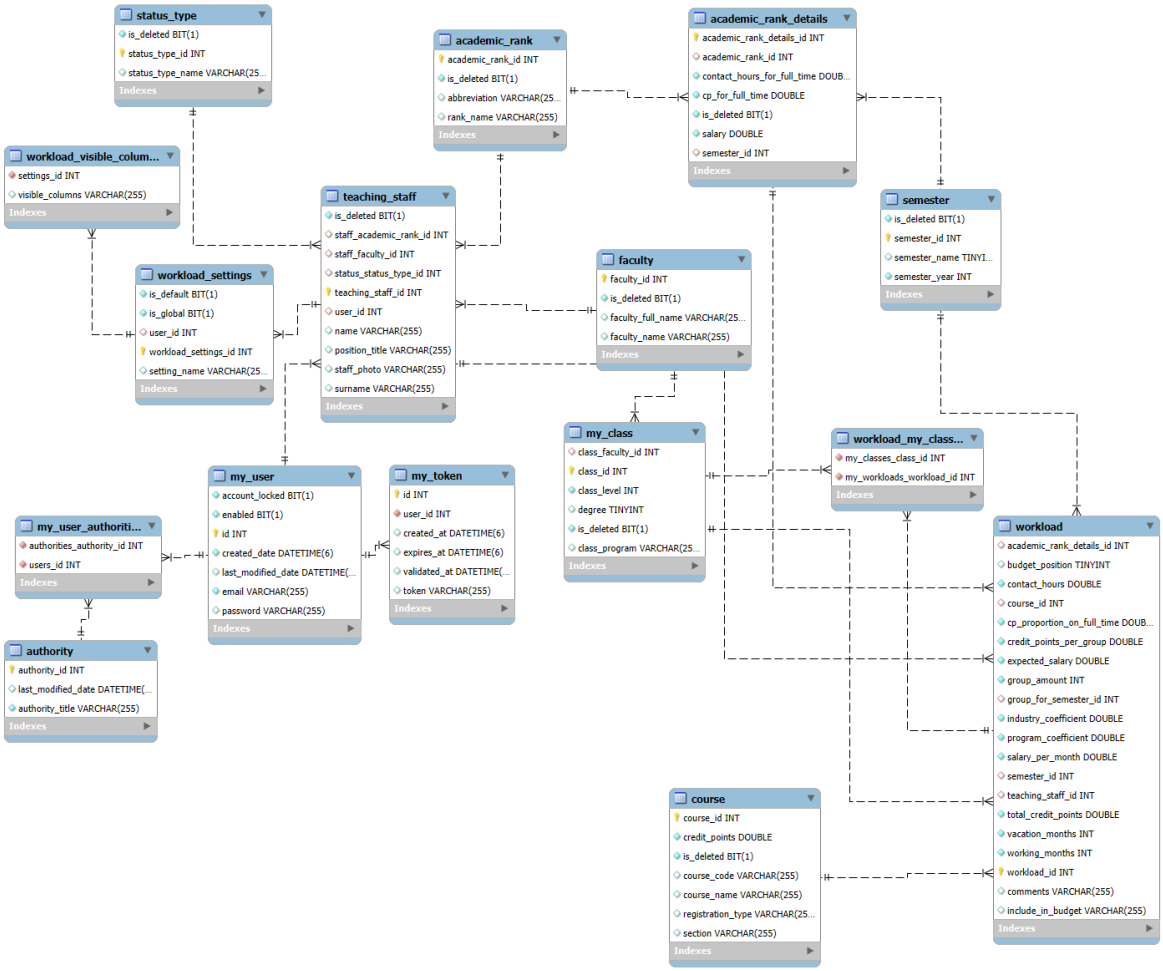
4.2. att. Labotais datu bāzes modelis

Lai viegli salīdzinātu un analizētu jaunās sistēmas datu modeli, izveidoju SQL skriptu, kurš pēc iespējas tuvāk esošajam datu modelim izvada datus. (skatīt pielikumā nr. 2). Labotais modelis tika piepildīts ar datiem, un izvade līdzinājās paraugam.

Taču aizmugursistēmas izveides laikā datu bāzes modelim, bija nepieciešamas izmaiņas un uzlabojumi, tāpēc tika veikta sekojoši uzlabojumi (4.3 att) kā rezultātā datubāzes modelis sastāv no 16 tabulām:

- Autorizācijas detaļas (my_user) tika savienots ar docētāju (teaching staff) tabulu, lai docētāji varētu piekļūt sistēmai un redzēt savu noslodzes plānu, attiecīgajā semestrī
- Noslodzes plānam ir nepieciešams sadalījums pa semestriem, tāpēc tika izveidota semestra (semester) tabula, kura ir savienota ar noslodzes plānu.

- Lai semestra sadalījums tiešām būtu loģisks, sadalīju amatu grupas tabulu (academic_rank) 2 daļās amatu grupu tabulā un amatu grupu detaļu (academic rank details) tabulā, amatu grupu tabulā atstājos informāciju par amata nosaukumu un abreviatūru, taču informāciju par kontaktstundām, algu, kredītpunktiem pilnai slodzei un galvenokārt atslēga uz semestra tabulu, tika ielikta jaunajā amatu grupu detaļu tabulā, tādejādi nodrošinot, ka ja jaunā semestrī ir cita alga, alga iepriekšējiem datiem nemainītos.
- Kā arī tika izveidota pieejas atslēgu tabula, kurā tiek saglabāti profila aktivizēšanas kodi, kuri tiek nosūtīti uz epastu nesen izveidotiem docētājiem.
- Tika izveidotas2 jaunas tabulas (workload_settings, workload_visible columns), lai nodrošinātu lietotājiem iespēju saglabāt tabulu iestatījumus savā profilā.
- Visām tabulām kuru dati ir izmantoti citā tabulā ar atslēgas palīdzību ieviesu drošu dzēšanu, jeb papildus kolonu, kura nosaka vai ieraksts ir dzēsts vai nē, tādejādi dati netiek dzēsti, bet gan paslēpti.
- Statusa tips (status type) tika noņemts no noslodzes plāna un savienots tieši ar docētāju (teaching staff)



4.3. att. Gala DB modelis.

5.2 Aizmugursistēmas modeļa slāņa izveide

Aizmugursistēma ļaus droši piekļūt datiem no datubāzes caur "REST API" galapunktiem. Lai izveidotu jaunajai sistēmai aizmugursistēmu izmantoju "Spring initializer" [4], kurš ļauj mājaslapā konfigurēt projektu – izvēlēties programmēšanas valodu, izvēlēties versiju, mainīt nosaukumu, pievienot bibliotēkas uc. Šim projektam izmantoju sekojošas bibliotēkas:

- "Spring-boot-starter-data-jpa" – ļauj veidot vai atjaunot savienoto datubāzi, ņemot vērā projekta klases.
- "Spring-boot-starter-mail" – ļauj savienot aizmugursistēmu ar epasta serveri, lai sūtītu epastus.
- "Spring-boot-starter-validation" – atvieglo un samazina kodu datu validācijai.
- "Mysql-connector-j" – ļauj savienot aizmugursistēmu ar datubāzi.
- "Lombok" – aizstāj bieži izmantotus un vieglus koda fragmentus ar anotāciju.
- "spring-boot-starter-security" – ievieš autorizāciju, ierobežo pieeju aizmugursistēmai uz noteiktām ip adresēm, kā arī ierobežo atļauto pieprasijumu veidus.
- "jjwt-api" – ļauj izveidot šifrētu atslēgu, kurā atrodas informācija par to cik ilgi viņa derīga, kam viņa pieder uc. Šo atslēgu pēctam var aizsūtīt lietotāja saskarnei, lai tā varētu autentificēt sevi aizmugursistēmai.
- "springdoc-openapi-starter-webmvc-ui" – grafiski attēlo aizmugursistēmas galapunktus, kā arī ļauj pārbaudīt, vai tie strādā.
- "opencsv" – bibliotēka, kas palīdz veidot, rediģēt, lasīt csv failus.
- "spring-boot-starter-thymeleaf" – strādā kopā ar aizmugursistēmu lai papildinātu html funkcionalitāti, piemēram, dot iespēju html kodā iespraust datus no aizmugursistēmas, šo izmantoju, lai veidotu paraugu epasta vēstulei.

Pēc projekta izveides konfigurēju aplikācijas iestatījumus, lai katru reizi palaižot aplikāciju, tā nemainītu datubāzes struktūru un tikai pārbaudītu esošo, tādejādi pasargājot datubāzes modeli no nejaušām izmaiņām izstrādes laikā, kā arī pievienoju adresi, caur kuru var piekļūt datubāzei, un iedevu autorizācijas informāciju. Parole un lietotājvārds aizmugursistēmai tiek padots palaišanas laikā.

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/mydatabase
    username: ${DB_NAME}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: validate
    show-sql: true
```

5.2.1 koda fragments. ***Spring*** **satvara konfigurācija**

Spring Boot aizmugursistēmai balstoties uz datubāzes modeli tika definēti 10 modeļi (5.2.1 tabula, 5.2.2 tabula, 5.2.3 tabula, 5.2.4 tabula, 5.2.5 tabula, 5.2.6 tabula, 5.2.7 tabula, 5.2.8 tabula, 5.2.9 tabula, 5.2.10 tabula).

Modeļa “Course” struktūra

5.2.1 tabula

Nr.	Nosaukums	Tulkojums	tips	Piezīmes
1	courseId	studiju priekšmeta identifikātors	Int	
2	courseCode	studiju priekšmeta kods	String	
3	courseName	studiju priekšmeta nosaukums	String	
4	creditPoints	studiju priekšmeta kredītpunkti	Double	
5	registrationType	reģistrācijas veids	String	
6	section	programmas daļa	String	
7	isDeleted	vai izdzēsts	Boolean	

Modeļa “AcademicRank” struktūra

5.2.2 tabula

Nr.	Nosaukums	Tulkojums	tips	Piezīmes
1	academicRankId	amatu grupas identifikātors	Int	
2	rankName	amatu grupas nosaukums	String	
3	abbreviation	abreviācija	String	
4	isDeleted	vai izdzēsts	Boolean	

Modeļa “AcademicRankDetails” struktūra

5.2.3 tabula

Nr.	Nosaukums	Tulkojums	tips	Piezīmes
1	academicRankDetailsId	amatu grupas detaļas identifikātors	Int	
2	cpForFullTime	kredītpunkti pilnai slodzei	Double	
3	salary	alga	Double	
4	contactHoursForFulltime	Kontaktstundas pilnai slodzei	Double	
5	semester	semestris	Semester	
6	academicRank	Amatu grupa	AcademicRank	
7	isDeleted	vai izdzēsts	Boolean	

Modeļa “Faculty” struktūra

5.2.4 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	facultyId	fakultātes identifikātors	Int	
2	facultyName	fakultātes abreviatūra	String	
3	facultyFullname	Fakultātes pilnais nosaukums	String	
4	isDeleted	vai izdzēsts	Boolean	

Modeļa “MyClass” struktūra

5.2.5 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	classId	kursa identifikātors	Int	
2	classLevel	kursa gads	Int	
3	classProgram	kursa programma	String	
4	degree	grāds	Degree	
5	classFaculty	kursa fakultāte	Faculty	savienots ar daudzi pret vienu attiecību
6	workloads	noslodzes	List<Workload>	savienots ar daudzi pret daudziem attiecību
7	isDeleted	vai izdzēsts	Boolean	

Modeļa “Semester” struktūra

5.2.6 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	semesterId	semestra identifikātors	Int	
2	semesterName	semestra nosaukums	SemesterEnum	
3	semesterYear	semestra gads	int	
4	isDeleted	vai izdzēsts	Boolean	

Modeļa “StatusType” struktūra

5.2.7 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	statusTypeId	statusa identifikātors	Int	
2	statusTypeName	statusa nosaukums	String	
4	isDeleted	vai izdzēsts	Boolean	

Modeļa “StatusType” struktūra

5.2.8 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	statusTypeId	statusa identifikātors	Int	
2	statusTypeName	statusa nosaukums	String	
4	isDeleted	vai izdzēsts	Boolean	

Modeļa “TeachingStaff” struktūra

5.2.9 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	teachingStaffId	docētāja identifikātors	Int	
2	name	vārds	String	
3	surname	uzvārds	String	
4	status	status	StatusType	
5	staffFaculty	fakultāte	Faculty	
6	staffAcademicRank	amatu grupa	AcademicRank	
7	positionTitle	amata nosaukums	String	
9	isDeleted	vai izdzēsts	Boolean	

Modeļa “Workload” struktūra

5.2.10 tabula

Nr.	Nosaukums	tulkojums	tips	Piezīmes
1	workloadId	noslodzes identifikātors	Int	
2	semester	semestris	Semester	
3	comments	piezīmes	String	
4	includeInBudget	iekļaut budžetā	String	
5	budgetPosition	budžeta pozīcija	BudgetPositions	
6	industryCoefficient	industrijas koeficents	Double	
7	vacationMonths	atvaļinājuma mēnešu skaits	int	
8	workingMonths	strādājamo mēnešu skaits	int	
9	groupAmount	grupu skaits	int	
10	contactHours	kontaktstundas	double	
11	creditPointsPerGroup	kredītpunkti grupai	double	
12	groupForSemester	kurss semestrim		
13	teachingStaff	docētājs	TeachingStaff	
14	course	studiju priekšmets	Course	
15	academicRankDetails	amatu grupas detaļas	AcademicRankDetails	

16	myClasses	kursi	List<MyClass>	savients ar daudzi pret daudziem attiecību
17	programCoefficient	programas koeficents	Double	automātiski aprēķināts ņemot vērā kursa grādu.
18	totalCreditPoints	kredītpunkti kopā	double	Automātiski aprēķināta ar formulu: grupu skaits * kredītpunkti grupai * programmas koeficents
19	expectedSalary	plānotā alga	double	Automātiski aprēķināta ar formulu: (atvaļinājuma mēnešu skaits + strādājamo mēnešu skaits) * alga mēnesī
20	cpProportionOnFullTime	slodzes daļa	double	Automātiski aprēķināta ar formulu: kredītpunkti kopā / kredītpunkti pilnai slodzei
21	salaryPerMonth	alga mēnesī	double	Alga * (slodzes daļa * industrijas koeficents)

Pēc ”Spring” konfigurācijas ir nepieciešams izveidot struktūru, kas nodrošina aizmugursistēmas funkcionalitāti. Šajā posmā tiek veidoti šādi komponenti:

- Datu modeļa slānis, kurš definē sistēmas datu struktūru un atspoguļo, kā dati tiks glabāti datubāzē, kā piemēram, *5.2.2 koda fragments.* Kurā redzami 4 mainīgie ar “semesterId” kā primāro atslēgu.

```
@Entity // anotācija, kura norāda ka šī klase ir datubāzes tabula ar
noklusējuma nosaukumu “Semester”
@Getter // pirms koda kompilācijas ģenerē klasei “getter” metodes
@Setter // pirms koda kompilācijas ģenerē klasei “getter” metodes
@AllArgsConstructor // pirms koda kompilācijas ģenerē klasei visu
argumentu konstruktoru
@Builder // lombok bibliotēka, kura atvieglo objektu izveidi.
@NoArgsConstructor// pirms koda kompilācijas ģenerē klasei bez-
argumentu konstruktoru

@ToString // pirms koda kompilācijas ģenerē klasei “toString()” metodi
kura pārvērš objektu teksta formā

public class Semester {
    @Id // marks semesterId as primary key
    @GeneratedValue(strategy = GenerationType.IDENTITY) // passes id
generation to database
    private int semesterId;
    private SemesterEnum semesterName;
```

```
    private int semesterYear;
    private boolean isDeleted;
}
```

5.2.2 koda fragments. ***"Semester" klases kods***

- Kontrolieri (Controller), kuri atbild par aizmugursistēmas publisko saskarni jeb "API" galapunktiem (endpoints). Tas saņem ienākošos pieprasījumus no klienta un novirza tos tālākai apstrādei, kā, piemēram, skatīt (pielikumu Nr. 3.), kur ir nodrošinātas objekta dzēšanas, atjaunošanas, izveides un iegūšanas metodes, kuras novirza tālāk uz servisa klasi, kurā tiek izpildīts pieprasījums.
- Objekta izveidošanas maketi, kuri definē datu struktūru, kāda ir nepieciešama, lai izveidotu vai atjaunotu resursus sistēmā, kā arī nodrošina datu validāciju un kļūdu ziņojumus.

```
public record SemesterRequest (
        @NotNull(message = "180")// mainīgais nevar būt tukšs, citādi
atgriež 180 http kļūdu
        SemesterEnum semesterName,
        @NotNull(message = "181")
        int year
){}
```

5.2.3 koda fragments. ***"Semester" izveides makets***

- Atbildes makets nosaka, kāda informācija tiks atgriezta klientam, pieprasot konkrēta objekta datus. Tas nodrošina strukturētu un paredzamu datu atgriešanu, kā arī ļauj viegli pievienot un noņemt papildus informāciju atbildes datiem.

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class SemesterResponse {
    private int semesterId;
    private SemesterEnum semesterName;
    private int year;
    private boolean isDeleted;
}
```

5.2.4 koda fragments. ***"Semester" atbildes makets***

- Kartētājs (Mapper) ir atbildīgs par datu transformāciju starp maketiem. Konkrēti, tas pārveido pieprasījumu maketus par datu bāzes objektiem un datu bāzes objektus uz atbildes maketu. Kā arī kartētājs ir lieliska vieta, kur aprēķināt automātiski aprēķināmos datus (*4.2.6 koda fragments*).

```
@Service
public class SemesterMapper {
    public Semester toSemester(SemesterRequest request){
        return Semester.builder()
```

```
                .semesterName(request.semesterName())
                .semesterYear(request.year())
                .isDeleted(false)
                .build();
    }
    public SemesterResponse toSemesterResponse(Semester semester){
        return SemesterResponse.builder()
                .semesterId(semester.getSemesterId())
                .semesterName(semester.getSemesterName())
                .year(semester.getSemesterYear())
                .isDeleted(semester.isDeleted())
                .build();
    }
}
```

5.2.5 koda fragments. ***"Semester" kartētājs***

```
public double getSalaryPerMonth(double salary, double
creditPointsOnFullTime, double industryCoefficient)
{
    return round(salary * creditPointsOnFullTime *
industryCoefficient,3);
}
```

4.2.6 koda fragments. ***"Workload" kartētājā izveidots automātisks aprēķins.***

- Servisa Slānis satur loģiku, kas nepieciešama, lai apstrādātu ienākošos pieprasījumus no kontroliera. Tas mijiedarbojas ar datu modeļa slāni. Kā, piemēram, "Semester" serviss (skatīt pielikumā Nr. 4), kur "Semester" objekti tiek saglabāti datubāzē, atjaunoti (atrod esošo ierakstu datubāzē un pārraksta to), izgūti un dzēsti, kā arī manipulē ar objekta datiem izmantojot kartētāju.

5.3 Autorizācijas izveide un drošības konfigurācija.

Ņemot vērā, ka sistēmā tiks izmantota Angular saskarne, aizmugursistēmas drošības nolūkos ir ieviesti šādi ierobežojumi:

- Atļauto IP adrešu skaits ir ierobežots līdz vienai.
- Pieprasījumu galvenēs ir atļautas šādas četras vērtības: ORIGIN, CONTENT_TYPE, ACCEPT, un AUTHORIZATION.
- Un pieprasījumu tipi ierobežoti uz "GET", "POST", "DELETE", "PUT" un "PATCH"

```
config.setAllowedOrigins(allowedOrigins); // atļautā ip adrese
config.setAllowedHeaders(Arrays.asList( // atļautās galvenes
        ORIGIN,
        CONTENT_TYPE,
        ACCEPT,
        AUTHORIZATION
));
config.setAllowedMethods(Arrays.asList( // atļautie pieprasījumu veidi
        "GET",
        "POST",
        "DELETE",
```

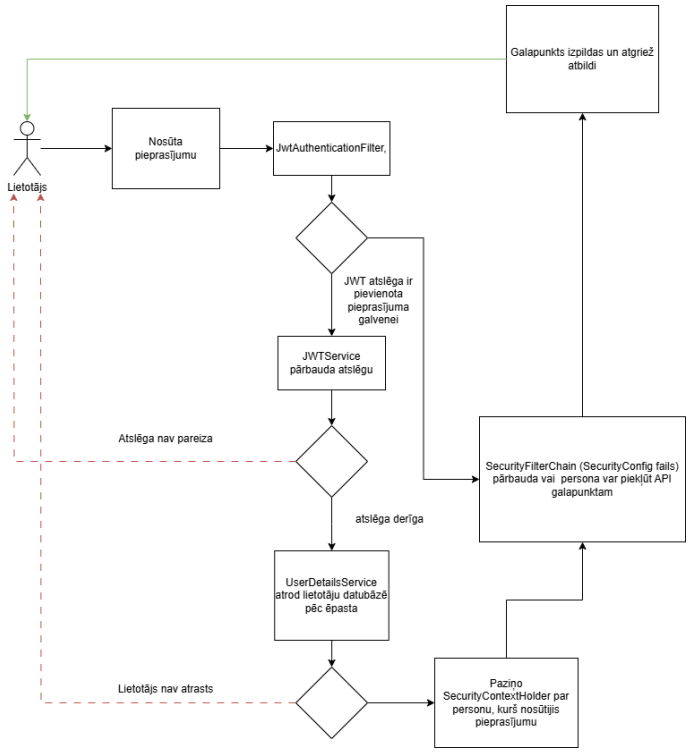
```
            "PUT",
            "PATCH"
));
```

5.3.1 koda fragments. ***Spring drošības konfigurācija***

Spring ietvaros, pirms katra pieprasījuma apstrādes, tas secīgi iziet cauri filtru virknei, kas veic pieprasījuma validāciju, iegūst informāciju no tā galvenēm un realizē dažādas darbības. Lai implementētu JWT autorizācijas mehānismu (attēls 5.3.1), tika izveidots izveidots filtrs "JwtAuthenticationFilter". Šis filtrs primāri pārbauda, vai pieprasījumam ir pievienota JWT atslēga. Gadījumā, ja atslēga netiek atrasta, filtra darbība tiek pārtraukta, nododot kontroli nākamajiem filtriem ķēdē. Šāda pieeja ir nepieciešama, lai nodrošinātu piekļuvi autorizācijas galapunktiem un ļautu lietotājiem veikt autentifikāciju.

Pēc tam, ja JWT atslēga ir atrasta, tiek izsaukts "JWTService", kura uzdevums ir veikt atslēgas validāciju. Veiksmīgas validācijas rezultātā, izmantojot "UserDetailsService", tiek veikta lietotāja meklēšana datubāzē pēc atslēgā iekļautās informācijas.

Visbeidzot, "JwtAuthenticationFilter" informē "SecurityContextHolder" par autentificēto lietotāju, kurš ir veicis pieprasījumu. Pēc šīs darbības tiek aktivizēti pārējie filtri, tostarp "SecurityFilterChain", kura galvenā funkcija ir pārbaudīt, vai autentificētajam lietotājam ir nepieciešamā loma, lai piekļūtu konkrētajam API galapunktam.



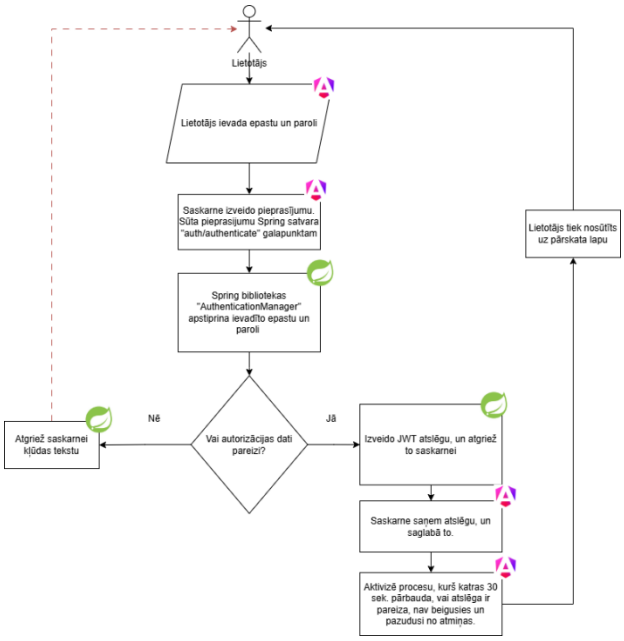
5.3.1 att. pieprasījumu pārbaudīšana

Autorizācijas procesa (5.3.2 att.) sākumā lietotājs ievada savu e-pastu un paroli saskarnē, pēc kā saskarne uz "/auth/authenticate" galapunktu nosūta pieprasījumu kopā ar datiem. Ievadītie dati nonāk pie "AuthenticationManager", kas veic to autentifikāciju.

Ja autentifikācijas procesā lietotājs netiek atrasts (piemēram, ievadīti nepareizi dati vai lietotājs nav reģistrēts), process tiek pārtraukts, un lietotājam tiek paziņots par kļūdu.

Savukārt, ja autentifikācija ir veiksmīga, tiek izsaukts “JWTService”. Šis serviss ģenerē jaunu JWT atslēgu, kurā tiek iekļauta informācija par lietotāju - tā vārds un uzvārds, e-pasts, lomas un atslēgas derīguma termiņš.

Ģenerētā JWT atslēga tiek nodota atpakaļ saskarnei. Kur, lai piekļūtu aizsargātiem galapunktiem saskarne šo atslēgu iekļaus katrā savā pieprasījumā un tā tiks pārbaudīta pēc 5.3.1 attēla. Kā arī saskarne katras 30 sekundes pārbauda vai atslēga joprojām ir derīga.

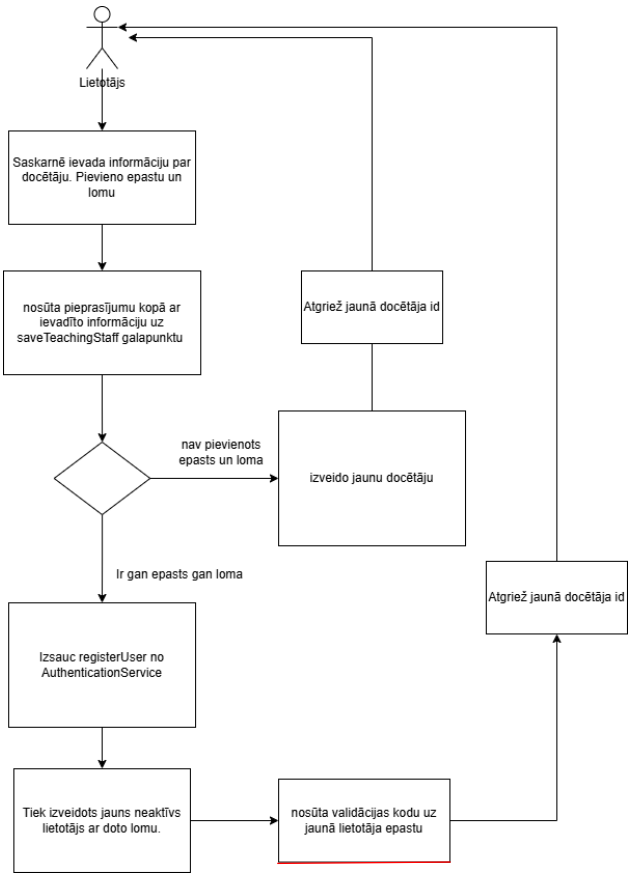


5.3.2 att. lietotāja autorizācija

Docētāju reģistrācija sistēmā (5.3.3 att.) sākas ar to ka administrators ievada informāciju par jauno mācībspēku, norādot vārdu, uzvārdu, amatu grupu un fakultāti taču ja vēlas lai personai tiktu izveidots profils nepieciešams ievadīt arī e-pastu un lomu sistēmā. Pēc tam lietotājs nosūta pieprasījumu kopā ar ievadīto informāciju uz “saveTeachingStaff“ galapunktu.

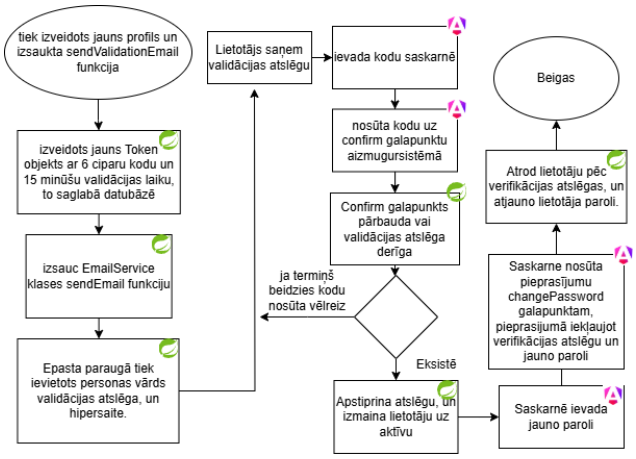
Sistēma pārbauda, vai pieprasījumā ir iekļauts gan e-pasts, gan loma. Ja kāda no šīm detaļām trūkst, tiek izveidots jauns docētāja ieraksts datubāzē, bet tālāk reģistrācijas process netiek turpināts, un sistēma atgriež jaunizveidotā docētāja ID.

Ja pieprasījumā ir gan e-pasts, gan loma, tiek izsaukts “registerUser” no “AuthenticationService”. Šis serviss izveido jaunu, neaktīvu lietotāju ar norādīto lomu sistēmā. Paralēli tam, uz jaunā lietotāja e-pastu tiek nosūtīts validācijas kods. Pēc veiksmīgas neaktīva lietotāja izveides, sistēma atgriež docētāja ID.



5.3.3 att. lietotāja autorizācija

Lai lietotāju aktivizētu (5.3.4), sistēma izveido 6 ciparu validācijas atslēgu, kuru saglabā datubāzē ar termiņu 15 minūtes, pēctam izsauc “EmailService”, kurš sagatavo ēpasta paraugu ievietojot tajā hipersaiti, validācijas atslēgu un lietotāja vārdu, un tad nosūta šo ēpastu lietotājam. Lietotājam saņemot validācijas e-pastu tas jāievada saskarnē, pēc kā saskarne nosūta pieprasījumu aizmugursistēmai apstiprināt validācijas atslēgu, ja atslēga pastāv taču vairs nav derīga, tiek atkārtoti nosūtīta atslēga, ja atslēga eksistē un ir derīga aizmugursistēma apstiprina lietotāju. Pēc koda ievades saskarnē tiek pieprasīts ievadīt savu paroli, kuru pēc tam kopā ar validācijas atslēgu nosūta aizmugursistēmai, kura nomaina paroli.



5.3.4 att. lietotāja aktivizācija

Sistēmā tika izveidotas 3 lomas:

- "ROLE_TEACHINGSTAFF" – piekļuve docētājiem;
  - redz ierobežota daudzuma datus, tikai par noslodzēm, kuras piesaistītas docētājam;
  - nedrīkst rediģēt, dzēst, izveidot.
- "ROLE_DIRECTOR" – piekļuve programmas direktoriem;
  - redz visus noslodzes ierakstus;
  - drīkst rediģēt, dzēst, izveidot.
- "ROLE_ADMIN" – piekļuve administratoriem;
  - visas darbības atļautas;

5.4 Noslodzes tabulas filtrācijas izveide.

Jaunajā sistēmā tāpat kā izklājlapās ir nepieciešama kolonu filtrācija un kārtošana. Lai funkcionalitāti nodrošinātu tika izveidots "API" galapunkts, kurš kā parametrus pieņem:

- Sort – nosaka kolonu, kura jakārto.
- Direction – nosaka kārtošanas virzienu.
- filtersJson – sastāv no kolonas nosaukumu, kuram ir filtra teksts, un operācijas veids.

Funkcionalitātes ieviešanai tiks izmantota bibliotēka "Specifications API", ar kuras palīdzību varam izgūt no datubāzes filtrētus datus, pievienojot specifikācijas "JPA" findAll funkcijai, kā piemēram, 5.4.1 koda fragmentā, kur vēlos izgūt visus noslodzes ierakstus, ar visiem filtriem, kuri atradās "filtersJson".

```
workloadRepo.findAll((root, query, criteriaBuilder) ->
  // šeit var veidot datu izgūšanai filtrus
  buildFilterPredicate(filters, root, criteriaBuilder, new
ArrayList<>()
// izsauc filtru veidotāja funkciju
);
```

5.4.1 koda fragments. ***Datu izgūšanas filtru iestatīšana***

Izsaucot funkciju "buildFilterPredicate" kura pārbauda, vai pieprasījuma ir pievienoti filtri, ja nav tad tiek izveidots tukšs filtrs un atgriezts "Specifications API" bibliotēkas "findAll" funkcijai, kura izvada visus ierakstus, taču ja pieprasījuma ir pievienoti filtri tiek izsaukta "applyFilters" funkcija, kura katram filtra ierakstam izsauc

“addPredicate” (skatīt 5.4.2 koda fragmentu) funkciju, kura balstoties uz filtrā esošo operātoru izveido filtru filtrā esošajai kolonai.

```
switch (operator) {
    case "contains":
        predicates.add(criteriaBuilder.like(
                criteriaBuilder.lower(path.as(String.class)),
                "%" + value.toLowerCase() + "%"));
        break;
    case "equals":
        predicates.add(criteriaBuilder.equal(
                criteriaBuilder.lower(path.as(String.class)),
                value.toLowerCase()));
        break;
    case "startsWith":
        predicates.add(criteriaBuilder.like(
                criteriaBuilder.lower(path.as(String.class)),
                value.toLowerCase() + "%"));
        break;
    case "endsWith":
        predicates.add(criteriaBuilder.like(
                criteriaBuilder.lower(path.as(String.class)),
                "%" + value.toLowerCase()));
        break;
}
```

5.4.2 koda fragments. ***Filtru veidošana***

5.5 Noslodzes tabulas iestatījumu izveide.

Noslodzes tabulā atrodas liels daudzums ar kolonām, kuras ne vienmēr būs lietotājam nepieciešamas, tāpēc tika izveidoti noslodzes iestatījumi, kuri ietver:

- iestatījumu nosaukumus;
- redzamās kolonas;
- vai ir noklusējuma;
- lietotāju kuram pieder iestatījums.

Savukārt saskarnē tika izveidota servisa komponente kuras uzdevums ir ielādēt visus autorizētās personas iestatījumus, izvēlēties noklusējuma iestatījumu un uzreiz to izvēlēties, kā arī sakarā ar to ka docētājiem kolonu skaits ir ievērojami mazāks, jo aizmugursistēma neatsūta visus datus, šī servisa komponente diferencē docētāja pieejamās kolonas no administratora pieejamām kolonām. Kad noklusējama iestatījumi izvēlēti noslodzes tabula paņem kolonas kuras nepieciešams rādīt no servisa komponentes un uzrāda tos.

5.6 Objektu veidošana ar csv tipa izklājlapu.

Lai atvieglotu migrēšanu no esošās sistēmas, nepieciešams jaunajā sistēmā ieviest objektu veidošanu ar csv tipa izklājlapu. Šīs funkcionalitātes nodrošināšanai tika izveidota

klase, kura kalpos kā makets csv tipa izklājlapas struktūrai, kā, piemēram, docētāju makets 5.5.1 koda fragmentā.

```
public class TeachingStaffCsvRepresentation {
    @CsvBindByName(column="email")
    private String email;
    @CsvBindByName(column="role")
    private String role;
    @CsvBindByName(column="statusId")
    int statusId;
    @CsvBindByName(column="staffFacultyId")
    int staffFacultyId;
    @CsvBindByName(column="staffAcademicRankId")
    int staffAcademicRankId;
    @CsvBindByName(column="name")
    String name;
    @CsvBindByName(column="surname")
    String surname;
}
```

5.5.1 koda fragments. ***Makets csv tipa izklājlapas struktūrai***

Pirms datu objektu izveides tiek inicializēts CsvToBean objekts, kas konfigurēts datu nolasīšanai atbilstoši 5.5.1. koda fragmentā definētajai struktūrai. Savukārt, strategy objekts nosaka, kā CSV faila kolonnas tiks saistītas ar izveidoto datu struktūru. Šajā gadījumā, tas panākts, sasaistot CSV faila pirmās rindas kolonnu nosaukumus ar TeachingStaffCsvRepresentation klases laukiem, tādējādi norādot bibliotēkai, kuram mainīgajam atbilstošā CSV kolonna jāpiešķir. Turklāt, konfigurācijā ir iekļauta tukšu rindu un rindu ar atstarpēm sākumā ignorēšana, kā arī datu atdalītājs ir definēts kā semikols (;), atbilstoši 5.5.2. koda fragmentam.

```
HeaderColumnNameMappingStrategy<TeachingStaffCsvRepresentation>
strategy = new HeaderColumnNameMappingStrategy<>();
strategy.setType(TeachingStaffCsvRepresentation.class);

new CsvToBeanBuilder<TeachingStaffCsvRepresentation>(reader)
        .withMappingStrategy(strategy)
        .withIgnoreEmptyLine(true)
        .withIgnoreLeadingWhiteSpace(true)
        .withSeparator(';')
        .build();
```

5.5.2 koda fragments. ***Datu ielasīšanas sagatavošana***

Lai uzlabotu lietotāju pieredzi, tika izstrādāts paraugs, ko lietotāji var lejupielādēt saskarnē. Šajā paraugā ir iekļauti kolonnu nosaukumi, datu ievades piemēri un aktuālie datubāzes objekti, piemēram, pieejamās fakultātes un amatu grupas. Katru reizi, kad lietotājs lejupielādē paraugu, pieejamās fakultātes un citi datubāzes objekti tiek atjaunināti, nodrošinot, ka paraugā vienmēr ir jaunākā informācija, kā tas ir ilustrēts koda fragmentā 5.5.3:

```
csvContent.append("# Pieejamie fakultasu id:\n");
    for (FacultyResponse faculty : faculties) {
      csvContent.append("# ").append(faculty.getFacultyId())
```

```
            .append(" - ").append(faculty.getFacultyName()).append("\n");
}
```

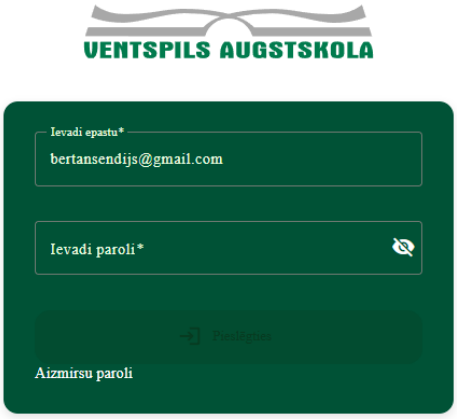
5.5.3 koda fragments. ***Datubāzes datu ievietošana paraugā.***

6 LIETOTĀJU SASKARNE

Lietotājam atverot sistēmu sev izvēlētā pārlūkprogrammā, tiks uzrādīta autorizācijas lapa, vai ja lietotājs jau iepriekš ir autorizējies, un nav manuāli izgājis tiks uzrādīta pārskata lapa.

6.1 Pieslēgšanās ekrāns

Lietotājs redz 2 laukus, kur pirmajā pieprasa ievadīt e-pastu, šis e-pasts tiek saglabāts lokāli un katru reizi nav jāievada un lauks, kur jāievada parole, parole ir paslēpta, taču to var uzrādīt spiežot uz actiņas ikonas. Pēc datu ievades abos laukos, pieslēgties poga izgaismojas, kā arī var nospiedz pogu “enter” lai autorizētos uzreiz. Kā arī autorizācijas lapā ir poga “aizmirsu paroli” kura aizved lietotāju uz paroles maiņas lapu. (*6.1.1 att.*)



6.1.1 att. autorizācijas lapa

Ja ievadīta nepareiza informācija tiek izvadīts kļūdas paziņojums – “Profils ar ievadīto ēpastu un vai paroli nēeksistē” (*6.1.2 att.*)

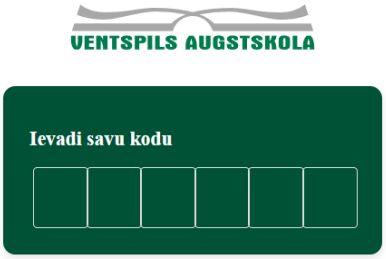
6.1.2 att. kļūdas paziņojums autorizācijas lapā

6.2 Paroles maiņas lapa

Lietotājam no autorizācijas lapas spiežot uz “Aizmirsu paroli” tiek atvērta paroles maiņas lapa. Šajā lapā atrodas 1 ievades lauks, kurā jāievada e-pasts, kad ievadīts strukturizēti pareizs e-pasts izgaismojas apstiprināt poga. (6.2.1 att.)

6.2.1 att. paroles maiņas lapa

Pēc apstiprināt pogas nospiešanas, lietotājs tiek novirzīts uz lapu, kur jāievada uz e-pastā nosūtītais kods, ievadot visus 6 ciparus, kods uzreiz tiek pārbaudīts tādejādi pogu nav jaspiež. (6.2.2 att.)



6.2.2 att. paroles maiņas lapa

Uz norādīto e-pastu nosūtīts e-pasts ar tēmu “Verifikācijas kods”, kurā norādīts personas Vārds un uzvārds, kods kuru var manuāli ievadīt paroles maiņas lapā (6.2.2 att.) un poga kura novirza uz sistēmu, kā arī automātiski ievada kodu. (6.2.3 att.)

6.2.3 att. verifikācijas koda e-pasts

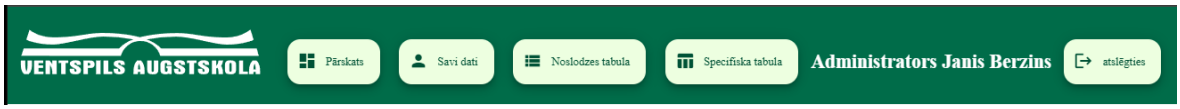
Pēc koda ievades lietotājs tiek novirzīts uz lapu kurā atrodas 2 lauki, kuros jāievada jaunā parole. (6.2.4 att.)

6.2.4 att. jaunās paroles izveides lapa

6.3 Pārskata lapa

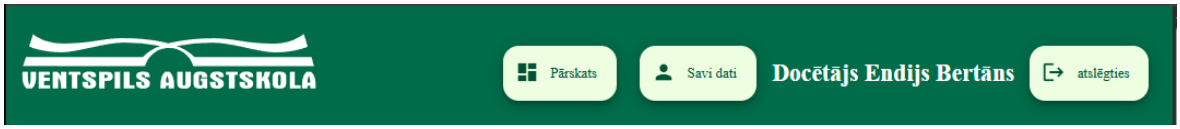
Pēc veiksmīgas autorizācijas lietotājs tiek autorizēts sistēmā un novirzīts uz pārskata lapu. Saskarnes galvenē administratoriem un direktoriem atrodas 5 pogas (6.3.1 att.):

- Pārskats poga novirza uz pārskata lapu
- Savi dati poga novirza uz noslodzes tabulas lapu, taču tabulā atrodas tikai dati, kuri attiecas uz autorizēto lietotāju.
- Noslodzes tabula poga novirza lietotāju uz noslodzes tabulu kurā atrodas dati par visiem docētājiem.
- Specifiskā tabula poga novirza uz lapu, kur atrodas dati no konkrētām tabulām datubāzē.
- Atslēgties poga atslēdz lietotāju no sistēmas.



6.3.1 att. saskarnes galvene

Docētājus arī pēc veiksmīgas autorizācijas novirza uz pārskata lapu, taču docētājiem galvenē atrodas tikai 3 pogas – pārskata, savu datu un atslēgties. (6.3.2 att.)

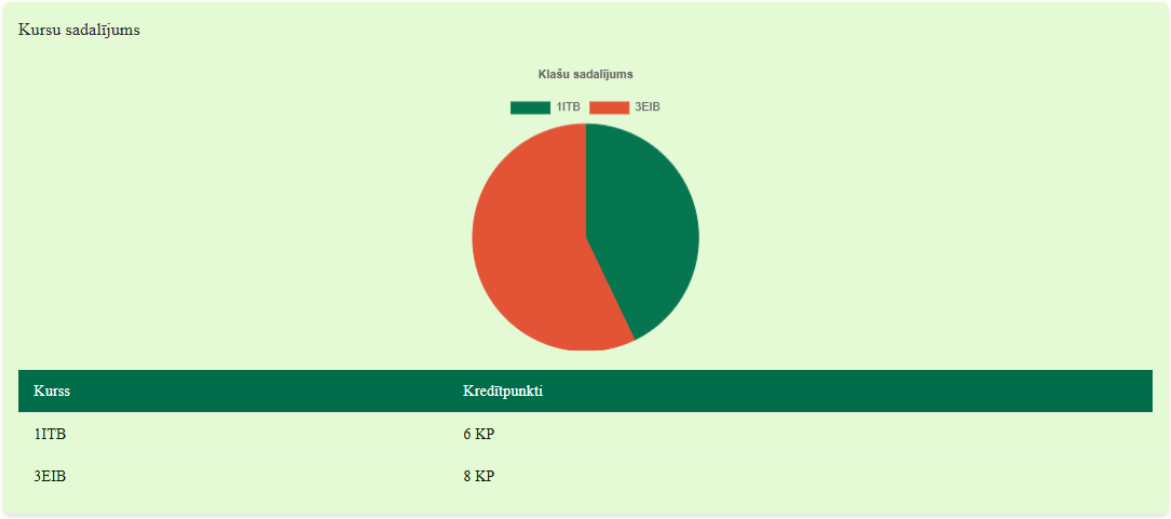


6.3.2 att. saskarnes galvene

Pārskata lapas sākumā atrodas 5 bloki, kuros atrodas informācija par slodžu skaitu, kontaktstundām, kredītpunktiem, algu un algu mēnesī. Kā arī opcija mainīt semestri.

6.3.3 att. 5 bloki pārskata sākumā

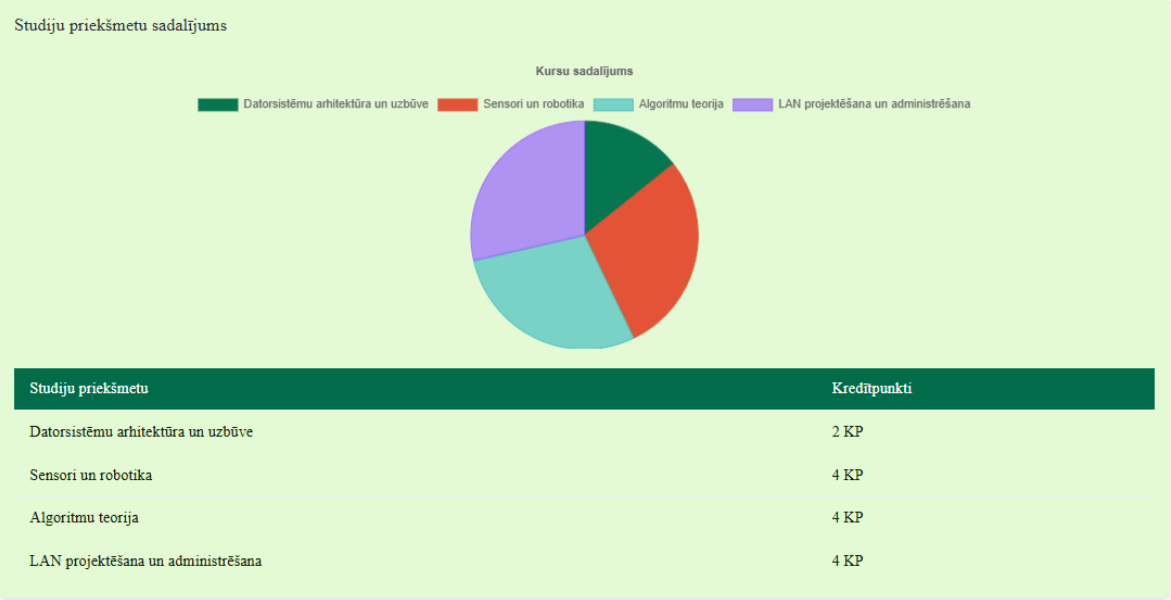
Pēc tam atrodas pīrāga diagramma, kurā uzzīmēta kredītpunktu attiecība starp kursiem, taču zem diagrammas tabula ar informāciju no kā sastāv diagramma. (6.3.3 attēls)



Kurss	Kredītpunkti
1ITB	6 KP
3EIB	8 KP

6.3.4 att. kredītpunktu attiecība starp kursiem

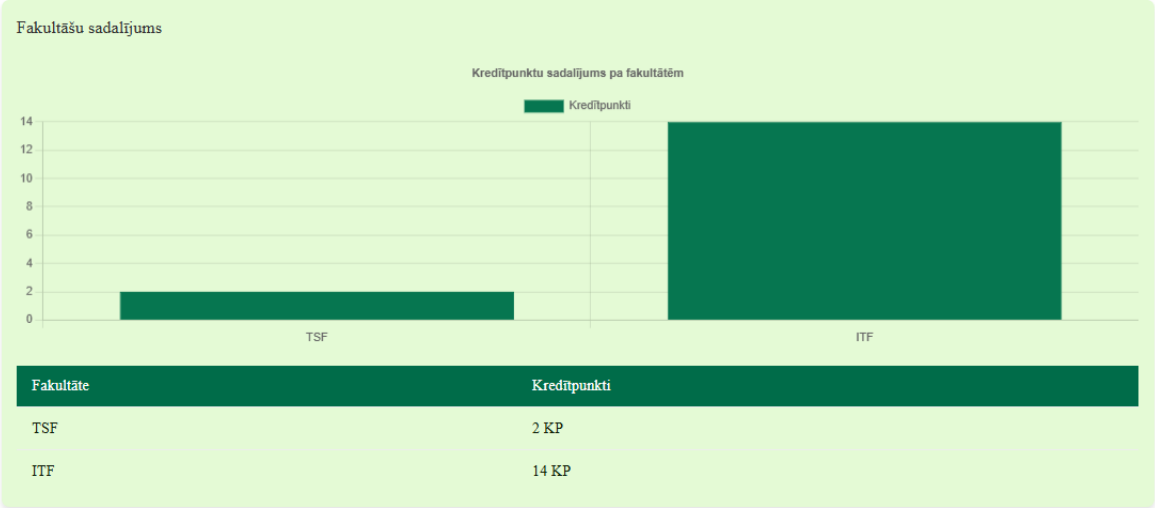
Pēc kredītpunktu attiecības starp kursiem atrodas kredītpunktu attiecība starp studiju priekšmetiem un arī tabula kura atrodas informācija no kā sastāv tabula.



Studiju priekšmetu	Kredītpunkti
Datorsistēmu arhitektūra un uzbūve	2 KP
Sensori un robotika	4 KP
Algoritmu teorija	4 KP
LAN projektēšana un administrēšana	4 KP

6.3.5 att. kredītpunktu attiecība starp studiju priekšmetiem

Taču pēdējā blokā atrodas kredītpunktu sadalījums starp fakultātēm, kurš attēlots stabiņu diagrammā un, protams, zem diagrammas atrodas tabula, kurā paskaidrots no kādas informācijas sastāv stabiņu diagramma.

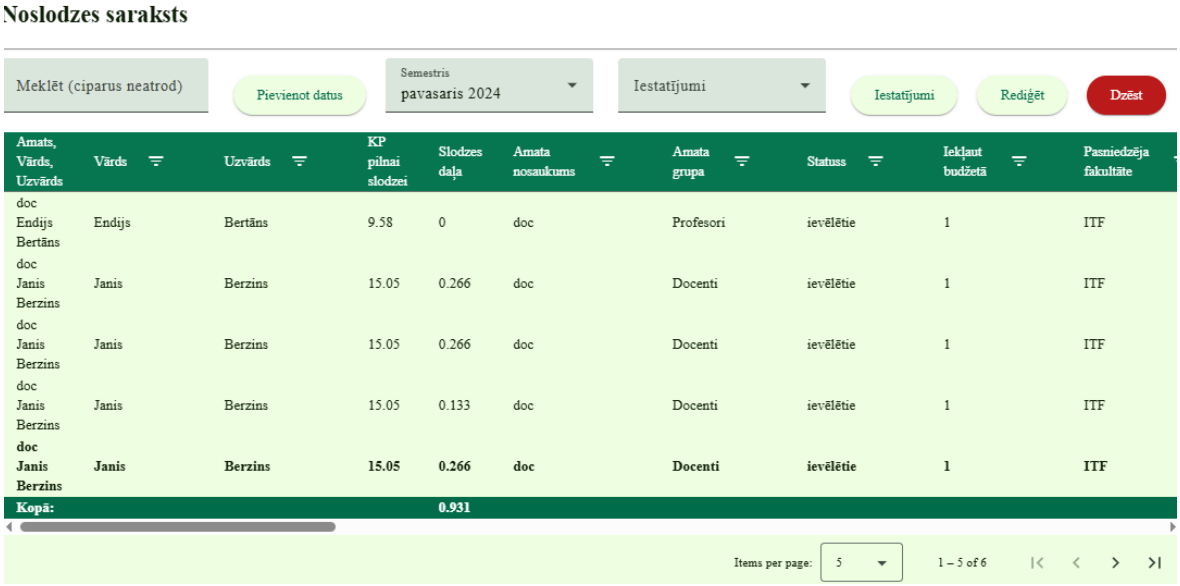


Fakultāte	Kredītpunkti
TSF	2 KP
ITF	14 KP

6.3.6 att. kredītpunktu attiecība starp fakultātēm

6.4 Noslodzes tabulas lapa

Administratora vai direktora skatā noslodzes tabulas lapā, tiek uzrādīta visa informācija izvēlētajā semestrī par visiem docētājiem, dati ir sadalīti lappusēs, tos var gan rediģēt, gan dzēst. Kā arī nodrošināta iespēja meklēt informāciju tabulā, mainīt semestri, izvēlēties sevis izveidotu iestatījumu, kurš paslēps norādītās kolonas. Taču tabulā iespējams specifiskai kolonai uzlikt kārtošanu. (6.4.1 att.)



6.4.1 att. noslodzes tabulas lapa

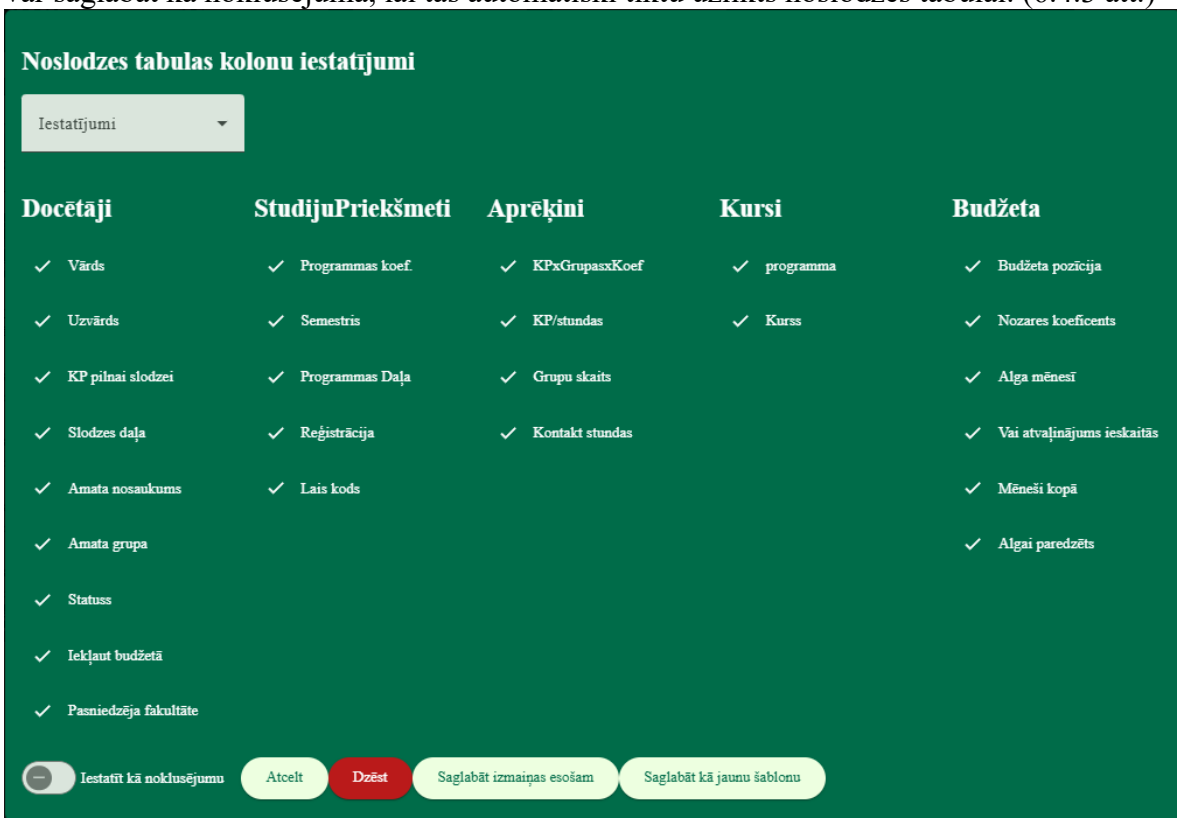
Kā arī katrai kolonai ir iespējams uzlikt teksta filtru ar pieejamām opcijām (6.4.2 att.):

- Ietilpst
- Vienāds
- Sākas ar

- Beidzas ar

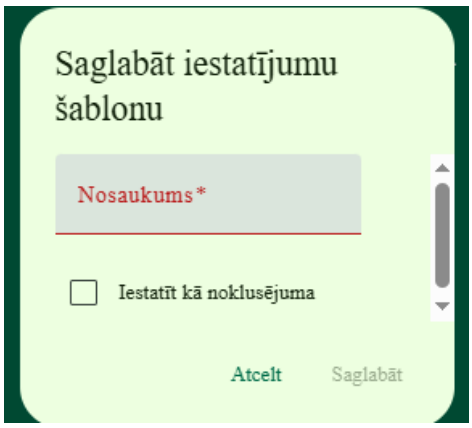
6.4.2 att. kolonu filtrēšana

Nospiežot Iestatījumi pogu, kā administrators vai direktors, lietotājs tiek novirzīts uz iestatījumu lapu, kurā iespējams mainīt kolonu redzamību, rediģēt saglabātos iestatījumus un dzēst tos. Kolonas sadalītas 5 sadaļās. Manevrēt starp saglabātajiem iestatījumiem var ar iestatījumu opciju, kura atrodas augšā labajā pusē. Kā arī iestatījumu var saglabāt kā noklusējuma, lai tas automātiski tiktu uzlikts noslodzes tabulai. (6.4.3 att.)



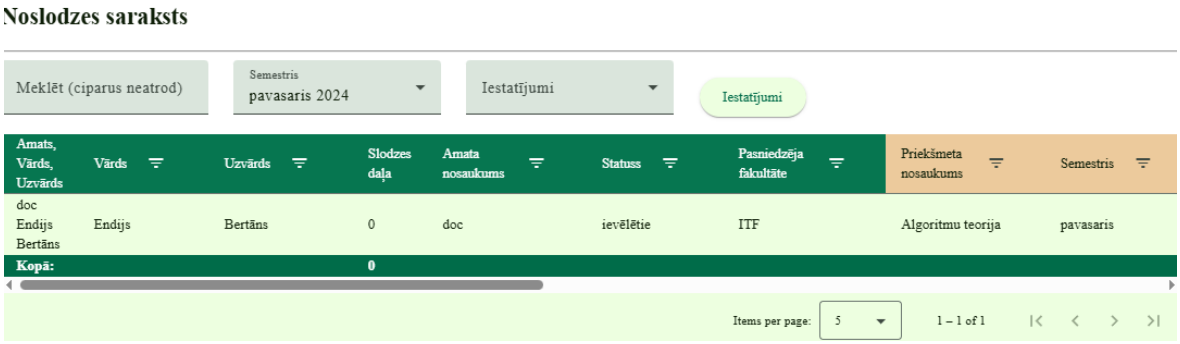
6.4.3 att. noslodzes tabulas kolonu redzamības iestatījumi

Saglabājot iestatījumu šablonu kā jaunu saskarnē atvērsies logs, kurā jaievada iestatījuma nosaukums.



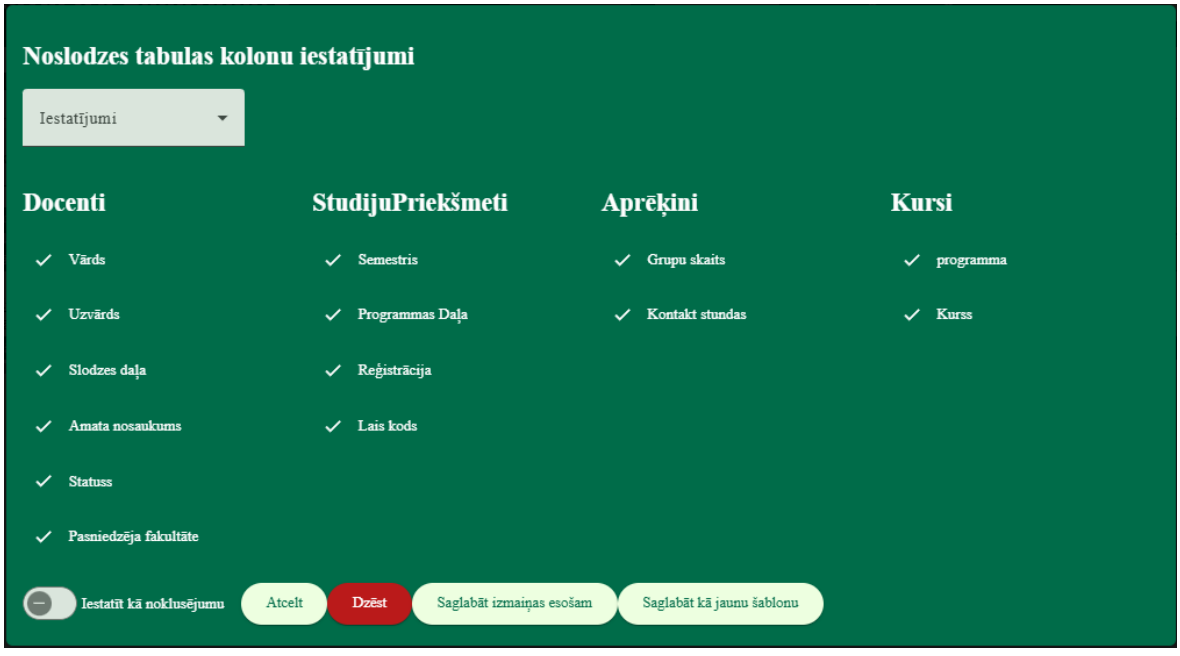
6.4.4 att. noslodzes tabulas kolonu redzamības iestatījumi

Savukārt autorizējoties kā docētājam, lietotājs drīkst skatīt iepriekšējus semestrus, un tabula uzrāda tikai autorizētā lietotāja datus, kā arī datu skaits no aizmugursistēmas ir samazināts. (6.4.5 att.)



6.4.5 att. noslodzes tabula docētājiem

Docētāji arī drīkst veidot, dzēst, rediģēt savu iestatījumu šablonu, taču ar mazāk kolonām. (6.4.6 att.)

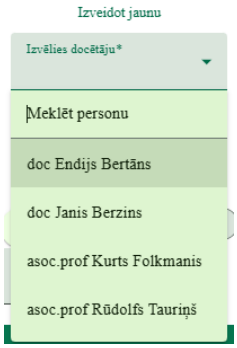


6.4.6 att. noslodzes tabulas kolonu redzamības iestatījumi docētājiem

Ar pogu izveidot jaunu, atveras izvēlne, kurā jāievadā docētājs, semestris, studiju priekšmeti, kursi, amats semestrim, kurss semestrim, nozares koeficients, grupu skaits, kredītpunktu skaits grupai, komentāri un budžeta pozīcija.

6.4.7 att. jaunas noslodzes izveide

Kā arī lai uzlabotu lietotāju pieredzi tika pievienota meklēšanas funkcija un iespēja izveidot jaunu docētāju, kurš automātiski tiek izvēlēts pēc izveides (6.4.8 att.)



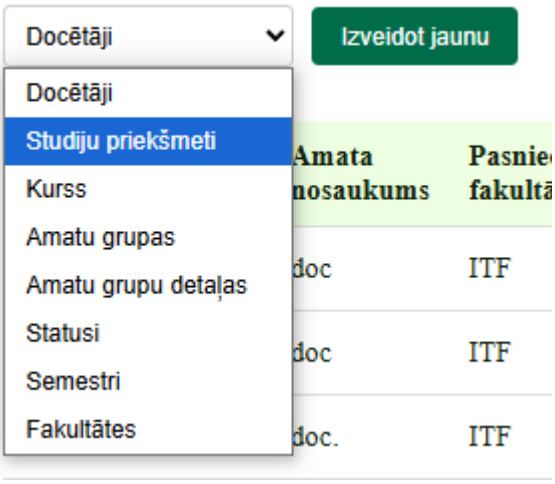
6.4.8 att. docētāju izvēlne ar meklēšanas opciju

Atzīmējot vairāk iestatījumu pievienojas papildu ievades lauki – iekļaut budžetā un atvaļinājuma mēneši. Kā arī ja lauciņā dati ievadīti nepareizi, vai atstāti tukši, lauks izgaismojas sarkans. (6.4.9 att.)

6.4.9 att. jaunas noslodzes izveide ar papildu laukiem

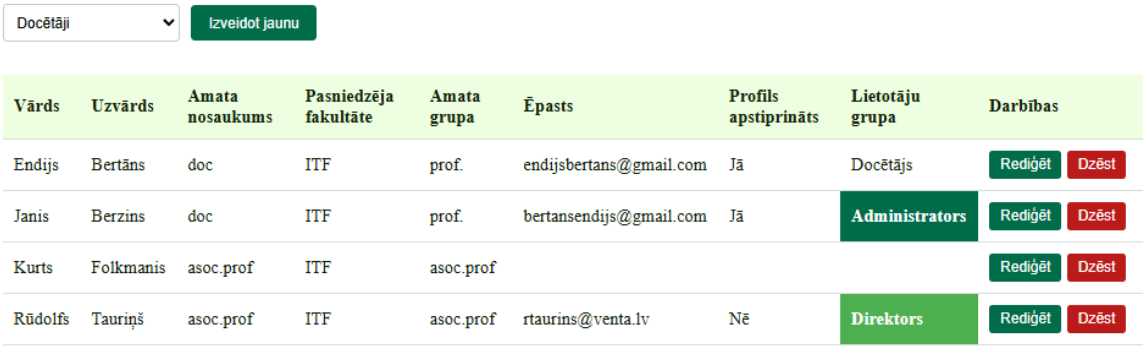
6.5 Datubāzes objektu lapa.

Atrodoties datubāzes objektu lapā, direktoram vai administratoram uzrādās izvelne, kurā iespējams mainīt tabulu un izveidot jaunu objektu. (6.5.1 att.)



6.5.1 att. izvelne starp datubāzes objektiem

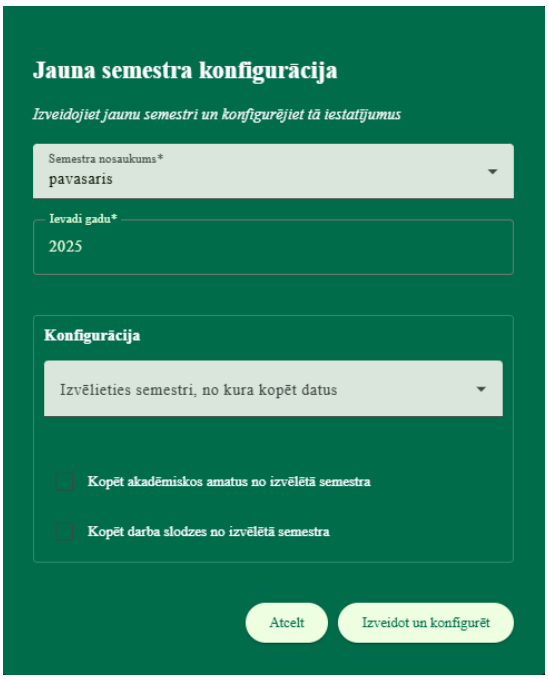
Objektu tabulā izvadīta visa informācija par objektu kā, piemēram, docētāju tabula, kur izvadīti visi docētāji, to loma un e-pasts, kā arī iespēja rediģēt un dzēst docētājus. (6.5.2. att.)



6.5.2 att. Objekta "Docētāji tabula"

6.6 Jauna semestra izveide

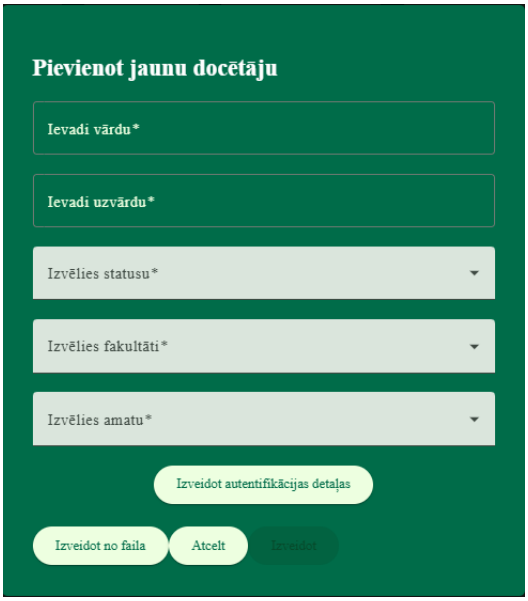
Lai atvieglotu jauna semestra iesākšanu, veidojot jaunu semestri ir nepieciešams ievadīt semestra nosaukumu un gadu, kad dati ievadīti ir iespējams kopēt amatu grupu detaļas un slodzes no jebkura cita semestra. (6.6.1)



6.6.1 att. Objekta "Docētāji tabula"

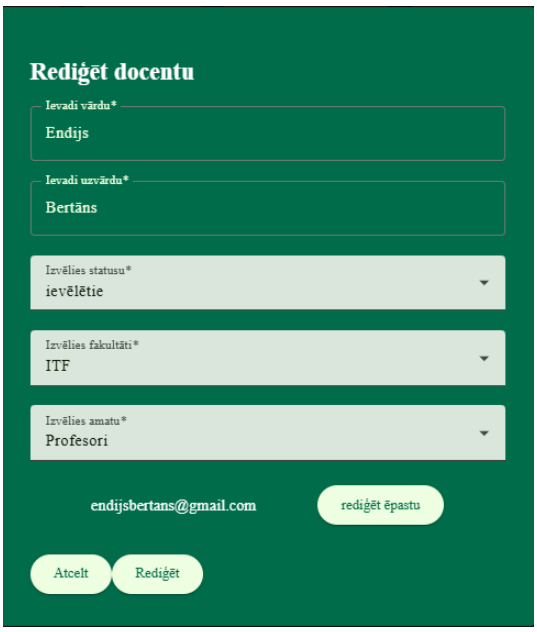
6.7 Objektu rediģēšana un izveide.

Objektus izveidot var no objektu datu lapas un noslodzes datu lapas, kad izveidot poga nospiesta atvērsies logs objekta izveidei kā, piemēram, jauna docētāja izveide, kur atrodas lauki kuros pieprasīta nepieciešamā informācija objekta izveidei, var būt lauki kuros manuāli jāieraksta dati, un lauki kuros atveras izvēlne, kurā jāizvēlas objekts no datubāzes, kuru piesaistīt docētājam (6.7.1 att.).



6.7.1 att. Objekta "Docētājs" izveide

Atzīmējot kādu objektu rediģēšanai atveras tāds pats logs kā objekta izveidei, taču lauki tiek automātiski aizpildīti. Mainot laukos aizpildīto informāciju un saglabājot izmaiņas objekts tiks rediģēts (6.7.2 att.).

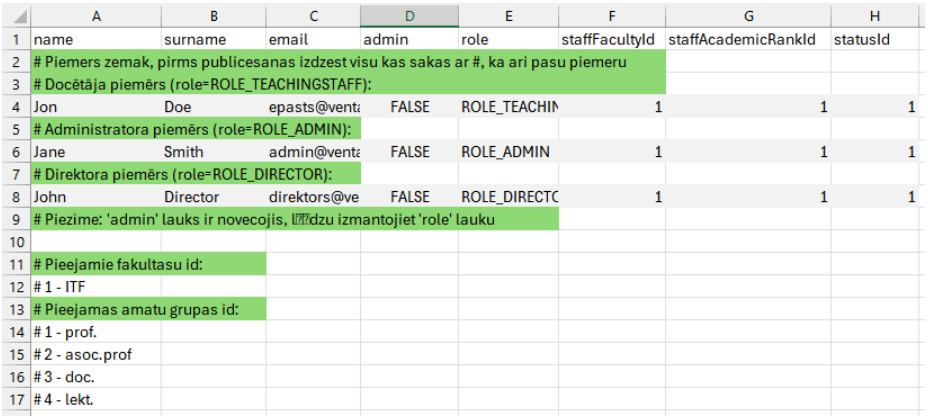


6.7.2 att. Objekta "Docētājs" rediģēšana

Lielu skaita objekta izveides atvieglošanai pie izveides pievienota poga ar tekstu "Izveidot no faila" (6.7.1 att.), šo pogu uzspiežot lietotājs tiek pārvirzīts uz lapu kurā pieejamas opcijas – izvēlēties failu, ielādēt paraugu, atcelt un izveidot.

6.7.3 att. Objekta "Docētājs" izveide no CSV failu tipa

Ielādējot paraugu tiek lejupielādēts fails, kuru var atvērt ar jebkuru izklājlapu lietotni, tajā atrodas galvene, kurā norādīts kāda informācija kolonā jāievada, tad daži piemēri datu ievadei, un no datubāzes izņemto objektu id. (*6.7.3 att.*)



	A	B	C	D	E	F	G	H
1	name	surname	email	admin	role	staffFacultyId	staffAcademicRankId	statusId
2	# Piemers zemak, pirms publicesanas izdzest visu kas sakas ar #, ka ari pasu piemeru							
3	# Docētāja piemērs (role=ROLE_TEACHINGSTAFF):							
4	Jon	Doe	epasts@venta	FALSE	ROLE_TEACHIN	1	1	1
5	# Administratora piemērs (role=ROLE_ADMIN):							
6	Jane	Smith	admin@venta	FALSE	ROLE_ADMIN	1	1	1
7	# Direktora piemērs (role=ROLE_DIRECTOR):							
8	John	Director	direktors@ve	FALSE	ROLE_DIRECTO	1	1	1
9	# Piezime: 'admin' lauks ir novecojis, l??dzu izmantojiet 'role' lauku							
10								
11	# Pieejamie fakultasu id:							
12	# 1 - ITF							
13	# Pieejamas amatu grupas id:							
14	# 1 - prof.							
15	# 2 - asoc.prof							
16	# 3 - doc.							
17	# 4 - lekt.							

6.7.3 att. Objekta "Docētājs" CSV faila paraugs

SECINĀJUMI UN PRIEKŠLIKUMI

Secinājumi

1. Veiktā esošās augstskolas slodzes pārvaldības sistēmas analīze atklāja tās vājās puses un ierobežojumus.
2. Bakalaura darba mērķis – izstrādāt jaunu noslodzes uzskaites sistēmu ar dažādām lomām - ir veiksmīgi sasniegts.
3. Izstrādātā sistēma nodrošina
   3.1. Automātiski aprēķinātus laukus.
   3.2. Ievadīto datu validāciju.
   3.3. Datu ievadīšanu no izklājlapas.
   3.4. Dažādas tiesības un datu pārskatus lietotāju lomām.
   3.5. Iespēju veidot jaunus semestrus, balstoties uz iepriekšējiem semestriem.
   3.6. Datu filtrēšanu, kārtošanu un sadalījumu pa lappusēm.
   3.7. Noslodzes tabulas redzamo kolonu iestatījumu saglabāšanu.
   3.8. Drošu datu dzēšanu.
4. Izstrādātā sistēma uzlabo slodzes pārvaldības procesu, samazinot kļūdu risku un veicinot datu pārredzamību.

Priekšlikumi

1. Veikt sistēmas pārbaudi reālā vidē iesaistot potenciālos lietotājus.
2. Integrācija ar citām augstskolas sistēmām.
3.

IZMANTOTĀS LITERATŪRAS UN AVOTU SARAKSTS

[1] DB-Engines Ranking [Tiešsaistē]. Pieejams: https://db-engines.com

[2] Stack overflow developer survey [Tiešsaistē]. Pieejams: https://survey.stackoverflow.co/2024/technology/

[3] Spring initializer [Tiešsaistē]. Pieejams: https://start.spring.io/

[4] Khawar Islam, Kamran Ahsan, S.A.K. Bari, Muhammad Saeed, and Syed Asim Ali, "Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer", *J. Basic Appl. Sci.*, vol. 13, pp. 481–490, Jan. 2017. [Tiešsaistē]. Pieejams: https://setpublisher.com/index.php/jbas/article/view/1719

[5] Bonteanu, A. M., & Tudose, C. (2024). Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA. *Applied Sciences*, *14*(7), 2743. [Tiešsaistē]. Pieejams: https://doi.org/10.3390/app14072743

[6] Dominik Choma, Kinga Chwaleba, Mariusz Dzieńkowski (20.12.2023) *THE EFFICIENCY AND RELIABILITY OF BACKEND TECHNOLOGIES: EXPRESS, DJANGO, AND SPRING BOOT* [Tiešsaistē]. Pieejams: 10.35784/iapgos.4279 [Skatīts 29.04.2025]

[7] Bogusz, D., Ciszewski, P., & Pańczyk, B. (2024). Performance analysis of web application client layer development tools using Angular, React and Vue as examples. *Journal of Computer Sciences Institute*, *32*, 223–230. https://doi.org/10.35784/jcsi.6299

[8] PAWEŁ DYMORA 1 , MIROSŁAW MAZUREK 2 , MARIUSZ NYCZ , Comparison of Angular, React, and Vue Technologies in the Process of Creating Web Applications on the User Interface Side (2023) [Tiešsaistē]. Pieejams: 10.15584/jetacomps.2023.4.21 [Skatīts Apr. 10, 2025]

[9] Spring dokumentācija (sadaļa identifikātori) [Tiešsaistē]. Pieejams: https://www.baeldung.com/hibernate-identifiers

[10] Spring dokumentācija (sadaļa objekti)[Tiešsaistē]. Pieejams: https://www.baeldung.com/jpa-entities

[11] Lombok bibliotēkas dokumentācija (sadaļa konstruktori)[Tiešsaistē]. Pieejams: https://projectlombok.org/features/constructor

PIELIKUMI

Pielikums Nr. 1

npk	Vārds	Uzvārds	Vārds,Uzvārds	Amats,Vārds,Uzvārds	KP pilnai slodzei	Slodzes daļa	Amata nosaukums	amata grupa	statuss	lekļautbudžetā	pasniedzēja fakultāte	semestris	Priekšmeta nosaukums
	Artūrs	Orbidāns	Artūrs Orbidāns	vieslekt. Artūrs Orbidāns	12	0	vieslekt.	lektori	neievēlētie	1	ITF	rudens	Elektronikas inženierijas projekts I
	Artūrs	Orbidāns	Artūrs Orbidāns	vieslekt. Artūrs Orbidāns	12	0	vieslekt.	lektori	neievēlētie	1	ITF	rudens	Mehānika

Programmas daļas studiju kursa kods	Priekšmeta KP	LAIS kods	reģistrācija	fakultāte	Programma	Grupa semestra grafikam	grupa		grupas skaits	KP skaits grupai	KPxGrupasx Koef	Komentāri	Akad.h./nedēļā	Programmas daļa
1	2	ETeh1001	automātiska	ITF	KNE	1KNE	1EIB+1KNE+LiepU	1KNE	0	1	0	praktiskie darbi		Nozares profesionālās specializācijas kursi
1	3	Meha2001	automātiska	ITF	KNE	1KNE	1EIB+1KNE	1KNE	0	1	0	praktiskie darbi		Nozares (profesionālās darbības jomas) teorētiskie pamatkursi un informācijas tehnoloģiju kur

Alga	Nozares koeficients	Alga mēnesī	Vai atvaļinājums ieskaitās	Mēnešu skaits	Algai paredzētais
900	1	0	0	5	0
900	1	0	0	5	0

```
      SELECT t.name as 'Vārds',
          t.surname as 'Uzvārds',
          CONCAT(t.name, ' ', t.surname) AS 'Vārds, Uzvārds',
          CONCAT(t.position_title, ' ', t.name, ' ', t.surname) AS
'Amats,Vārds, Uzvārds',
          aStaff.cp_for_autumn AS 'KP pilnai slodzei',
          w.cp_proportion_on_fulltime AS 'Slodzes daļa',
          t.position_title AS 'Amata nosaukums',
          aStaff.rank_name AS 'amata grupa',
          s.status_type_name AS 'statuss',
          w.include_in_budget AS 'Iekļaut budžetā',
          fstaff.faculty_name AS 'pasniedzēja fakultāte',
          w.semester AS 'semestris',
          co.name AS 'Priekšmeta nosaukums',
          w.credit_points_per_hour AS 'Progr. koef. KP/stundas',
          co.credit_points AS 'Priekšmeta kp',
          w.credit_points_per_group AS 'KP skaits grupai',
          w.group_amount AS 'grupu skaits',
          w.contact_hours AS 'Kontaktstundas',
          co.course_id AS 'LAIS kods',
          co.section AS 'Programmas daļa',
          co.registration_type AS 'Reģistrācija',
          fCourse.faculty_name AS 'Priekšmeta fakultāte',
          fClass.faculty_name AS 'kursa fakultāte',
          program AS 'Programma',
          co.study_level AS 'Studiju līmenis',
          w.group_for_semester AS 'Grupa semestra grafikam',
          -- grupa MANY TO MANY????
          GROUP_CONCAT(class.name SEPARATOR ', ') AS classes,
          w.comments AS 'Komentāri',
          w.budget_position AS 'budžeta pozīcija',
          GROUP_CONCAT(class.student_amount SEPARATOR ', ') AS 'Studentu
skaits',
          aStaff.salary AS 'Alga',
          w.industry_coefficiant AS 'Nozares koef.',
          w.salary_per_month AS 'Alga mēnesī',
          w.include_in_budget AS 'Vai atvaļinājums ieskaitās',
          w.month_amount AS 'Mēnešu skaits',
          w.expected_salary AS 'Algai paredzētais'
```

```
	FROM workload AS w
	JOIN teaching_staff AS t ON w.teaching_staff_id = t.teaching_staff_id
	JOIN status_type AS s ON s.status_type_id = w.status_type_id
	JOIN course AS co ON co.course_id = w.course_id
	JOIN academic_rank AS aCourse ON aCourse.academic_rank_id =
co.necessary_academic_rank_id
	JOIN academic_rank AS aStaff ON aStaff.academic_rank_id =
w.academic_rank_id
	JOIN faculty AS fStaff ON fstaff.faculty_id = t.staff_faculty_id
	JOIN faculty AS fCourse ON fCourse.faculty_id =
co.necessary_academic_rank_id
	JOIN class_junction AS classId ON classId.workload_junction_id =
w.workload_id
	JOIN class AS class ON class.class_id = classId.class_junction_id
	JOIN faculty AS fClass ON fClass.faculty_id = class.class_faculty_id
	GROUP BY
	w.workload_id
```

```
@RestController
@RequiredArgsConstructor // inicializē "final" mainīgos
@RequestMapping("semester")
public class SemesterController {
    private final SemesterService semesterService;
    private final AcademicRankDetailsService
academicRankDetailsService;
    private final WorkloadService workloadService;
    @PostMapping
    public ResponseEntity<Integer> saveSemester(
            @Valid @RequestBody SemesterRequest request){
        return ResponseEntity.ok(semesterService.save(request));
    }
    @PatchMapping("{semesterId}")
    public ResponseEntity<Integer> updateSemesterById(
            @PathVariable Integer semesterId,
            @Valid @RequestBody SemesterRequest request
    ){
        return ResponseEntity.ok(semesterService.update(semesterId,
request));
    }
    @DeleteMapping("{semesterId}")
    public ResponseEntity<Integer> deleteSemesterById(
            @PathVariable Integer semesterId
    ){
        return ResponseEntity.ok(semesterService.delete(semesterId));
    }
    @GetMapping("{semesterId}")
    public ResponseEntity<SemesterResponse> findSemesterById(
            @PathVariable Integer semesterId){
        return ResponseEntity.ok(semesterService.findById(semesterId));
    }
    @GetMapping
    public ResponseEntity<List<SemesterResponse>> findAllSemesters(){
        return ResponseEntity.ok(semesterService.findAllStatusTypes());
    }
}
```

```
@Service
@RequiredArgsConstructor
public class SemesterService {
    private final SemesterMapper semesterMapper;
    private final ISemesterRepo semesterRepo;

    public Integer save(SemesterRequest request) {
        Semester semester = semesterMapper.toSemester(request);
        return semesterRepo.save(semester).getSemesterId();
    }
    public Integer update(Integer semesterId, @Valid SemesterRequest
request) {
        Semester existingSemester =
findSemesterFromResponseId(semesterId);
        Semester updatedSemester = semesterMapper.toSemester(request);


updatedSemester.setSemesterId(existingSemester.getSemesterId());

        return semesterRepo.save(updatedSemester).getSemesterId();
    }

    public Semester findSemesterFromResponseId(int id) {
        return semesterRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Semester with
id: " + id + " not found."));
    }
    public SemesterResponse findById(Integer statusTypeId) {
        return semesterRepo.findById(statusTypeId)
                .map(semesterMapper::toSemesterResponse)
                .orElseThrow(()-> new EntityNotFoundException("Semester
with id: " + statusTypeId + " not found"));
    }
    public List<SemesterResponse> findAllStatusTypes() {
        List<Semester> semesters =
semesterRepo.findByIsDeletedFalseOrderBySemesterYearDesc();
        return semesters.stream()
                .map(semesterMapper::toSemesterResponse)
                .toList();
    }

    public Integer delete(Integer semesterId) {
        Semester semester = findSemesterFromResponseId(semesterId);
        semester.setDeleted(true);
        semesterRepo.save(semester);
        return semesterId;
    }
}
```

GALVOJUMS

Ar šo es, Endijs Bertāns, galvoju, ka šis bakalaura darbs ir manis paša patstāvīgi izpildīts oriģināls darbs. Visi informācijas avoti, kā arī no tiem ņemtie dati un definējumi ir norādīti darbā. Šis darbs tādā vai citādā veidā nav iesniegts nevienai citai pārbaudījumu komisijai un nav nekur publicēts.

Esmu informēts (-a), ka mans bakalaura darbs tiks ievietots un apstrādāts Vienotajā datorizētajā plaģiāta kontroles sistēmā plaģiāta kontroles nolūkos.

202__. gada _____ _____ _____
(paraksts)

Es, Endijs Bertāns, atļauju Ventspils Augstskolai savu bakalaura darbu bez atlīdzības ievietot un uzglabāt Latvijas Nacionālās bibliotēkas pārvaldītā datortīklā Academia (www.academia.lndb.lv), kurā tie ir pieejami gan bibliotēkas lietotājiem, gan globālajā tīmeklī tādā veidā, ka ikviens tiem var piekļūt individuāli izraudzītā laikā, individuāli izraudzītā vietā.

Piekrītu _____

Nepiekrītu _____

202__. gada _____ _____