**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. 00000000000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(paraksts)

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(paraksts)

Zinātniskais vadītājs \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ieņemamais amats, zinātniskais nosaukums, vārds, uzvārds)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(paraksts)

**ANOTĀCIJA**

**Darba nosaukums:** Jaunas attēlu segmentācijas metodes izstrāde.

**Darba autors:** Endijs Bertāns

**Darba vadītājs:** dr. math.Oskars Jansons

**Darba apjoms:** 3 lpp., 0 tabulas, 0 attēli, 0 bibliogrāfiskie norādes, 0 pielikumi.

**Atslēgas vārdi:** ATTĒLU SEGMENTĀCIJA, ALGORITMI, ATTĒLU UZLABOŠANA.

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, ar iespēju to izvietot AWS infrastruktūrā.

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

Saturs

[Ievads 4](#_Toc195043780)

[2. Esošās sistēmas analīze 5](#_Toc195043781)

# Ievads

Mūsdienās valsts un privātām iestādēm arvien būtiskāka kļūst efektīva, viegla un droša procesu pārvaldība, lai šos procesus atvieglotu, paātrinātu, un veiktu datu analīzi. 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 pārmērīgas darbinieku slodzes. Tā rezultātā palielinās kļūdu risks, un palielinās administratīvās darbības izmaksas. Šīs problēmas risināšanai ir plānots izstrādāt docētāju noslodzes pārvaldības sistēmu, kuras mērķis ir automatizēt semestra plānu veidošanas procesu, nodrošināt efektīvāku informācijas apmaiņu starp sistēmas lietotājiem, piemēram, docētājiem un dekāniem, kā arī nodrošināt datu saglabāšanu vienā sistēmā, lai varētu viegli pārskatīt citu mācību gadu datus.

Darba mērķis ir docētāju noslodzes pārvaldības sistēmas izveide, kura tiktu ieviesta vismaz vienā fakultātē. Šī sistēma sniegs iespēju digitalizēt un automatizēt semestra plānu veidošanas procesu, tādējādi ievērojami atvieglojot administratīvo darbu un samazinot kļūdu iespējamību.

Lai sasniegtu izvirzīto mērķi, pētījumā tiks veikti sekojoši uzdevumi:

* Analizēt esošo semestra plānu izveides procesu, izpētīt tā trūkumus un identificēt uzlabojumu iespējas.
* Modelēt un izstrādāt datubāzi, pēc esošo excel izklājlapu, piemērmiem, kas spētu efektīvi un droši uzglabāt nepieciešamo informāciju.
* Analizēt dažādas datu bāzes, lai secinātu kuru šai sistēmai izmantot būs vispiemērotāk.
* Realizēt projektēto datubāzi, kas tiks integrēta sistēmas funkcionalitātē.
* Manuāli ievadīt paragudatus no izklajlapām, lai pārliecinātos, ka tiek saglabāta esošā funkcionalitāte un visiem datiem no izklajlapas ir vieta datubāzē.
* Analizēt dažādas aizmugursistēmas (BACKEND??? Ņemts no tildes), lai noteikta, kura būs vislabāk piemērotākā, šai sistēmai.
* Izveidot aizmugursistēmu saskarnei ar datubāzes tabulām - datu izveidei, lasīšanai, atjaunošanai un dzēšanai.
* Analizēt pieejamos drošības risinājumus, izvēlētajām sistēmām.
* Ieviest sistēmā drošības risinājumus.
* Analizēt dažādus risinājumus, sistēmas saskarnes izveidei.
* 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, 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, kā arī visaptverošu testēšanu, lai nodrošinātu sistēmas uzticamību un lietojamību reālās darba situācijās, tādejādi tiks nodrošināta administratīvo procesu digitalizācija un automatizācija Ventspils Augstskolā.

# 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, vēlos piebilst velreiz, ka 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 jataisa 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, informācijas fragmentāciju un nepietiekamu validācijas mehānismu. Automatizēta, centralizēta un lietotājam draudzīga sistēma varētu ievērojami uzlabot procesa caurspīdību, samazināt kļūdu iespējamību un optimizēt resursu izmantošanu, tādējādi nodrošinot efektīvāku administratīvo procesu izpildi.

# pRASĪBU ANALĪZE.

Pamatojoties uz esošās sistēmas analīzi, ir iespējams identificēt un formulēt galvenās prasības jaunās centralizētās sistēmas izstrādei. 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.

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

Svarīgs nosacījums ir spēja ātri atjaunot datu kopas katra mācību gada sākumā, izmantojot iepriekšējo gadu informāciju kā bāzi. Automatizēta datu migrācija un integrācija ar vecajām sistēmām būtiski samazinās laika un cilvēkresursu patēriņu, kā arī minimizēs pārejas riskus.

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 pazaudēšanas un uzlabos sistēmas uzticamību.

Analizējot pašreizējo mācību gada plāna izveides procesu, 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.

# Tehnoloģiju salīdzinājums un izvēle sistēmas izstrādei

Izstrādājamās sistēmas darbībai būtiska komponente ir uzticama un efektīva datubāzes pārvaldības sistēma (DBVS). Lai identificētu populārākās un plašāk pielietotās datubāzes, tika izmantots **DB-Engines Ranking** (<https://db-engines.com/en/ranking>), kas apkopo dažādu datubāzu popularitātes rādītājus, balstoties uz vairākām metrikām, piemēram, meklējumu skaitu, diskusiju aktivitāti, tehnisko dokumentāciju un darba piedāvājumiem.

Saskaņā ar šī reitinga datiem (2025. gada sākumā) populārākās datubāzes ir:

Oracle

MySQL

Microsoft SQL Server

PostgreSQL

MongoDB

Neraugoties uz Oracle datubāzes popularitāti un izplatību korporatīvajā vidē, tās ieviešana prasa ievērojamus resursus un licenciēšanas izmaksas, kas šajā projektā nav pamatoti. Līdzīga situācija attiecas uz Microsoft SQL Server, kurš gan nodrošina plašu funkcionalitāti, bet ir komerciāls risinājums. MongoDB kā NoSQL datubāze labi piemērota dokumentu tipa datiem, tomēr šajā projektā pamatā ir strukturēti, relāciju dati, kas padara relāciju datubāzes piemērotākas.

Mūsdienu tīmekļa lietojumprogrammu izstrādē aizmugursistēmas izvēle ir īpaši svarīga, lai nodrošinātu drošu un efektīvu datu lasīšanu, rakstīšanu, rediģēšanu un dzēšanu. Lai gan nav viena universāla reitings, kas precīzi atspoguļotu backend ietvaru popularitāti 2025. gadā, izplatītākie ietvari ir Spring Boot (Java), Express.js (Node.js), Django (Python), Ruby on Rails (Ruby) un ASP.NET Core (C#). Katrai no šīm tehnoloģijām piemīt savi unikālie aspekti, kas gan pozitīvi, gan negatīvi ietekmē to piemērotību atkarībā no projekta prasībām.

Spring Boot ir Java ietvars, kas ir izstrādāts, lai vienkāršotu sarežģītu, uzņēmuma līmeņa lietojumprogrammu izstrādi. Tā automatizētā konfigurācija un iebūvētās funkcionalitātes, kā drošības un lietotāju bibliotēkas, padara to par ļoti piemērotu risinājumu projektiem, kas prasa augstu veiktspēju un mērogojamību. Spring Boot nodrošina stabilitāti un drošību. Taču salīdzinot ar vieglākiem ietvariem, Spring Boot var prasīt lielākus sistēmas resursus, it īpaši ja tas tiek pielietots mazāku projektu kontekstā.

Izstrādājot RESTful API, Express.js piedāvā ātru un vienkāršu risinājumu JavaScript vidē. Šī ietvara elastība ļauj izstrādātājiem brīvi pielāgot funkcijas atbilstoši specifiskām prasībām, taču šī vienkāršība var radīt problēmas, mērogojot lielākus projektus. Piemēram, struktūras un uzturēšanas jautājumi var prasīt papildu rūpīgu organizāciju, un maza iebūvēta drošība nozīmē, ka izstrādātājiem būs jāiegulda papildu laiks, lai pielāgotu projektu drošībai.

Python balstītais Django ietvars izceļas ar savu "batteries-included" filozofiju, kurā integrētas autentifikācijas sistēmas, ORM un aizsardzības protokoli, kas padara to par drošu un mērogojamu risinājumu. Django piedāvā labu veiktspēju un vienkāršu lietošanas uzsākšanu, taču noteiktās situācijās tā strukturētā pieeja var ierobežot izstrādātāja brīvību, īpaši ja ir nepieciešams unikāls risinājums.

Ruby on Rails, balstīts uz "konvencijas pār konfigurāciju" principu, paātrina izstrādes procesu un vienlaikus sniedz spēcīgu kopienas atbalstu, kas ir vērtīgs lielāku projekttu ietvaros. Tomēr, izmantojot Ruby on Rails, var rasties veiktspējas problēmas, it īpaši intensīva slodzes situācijās, kā arī noteiktas konvencijas var ierobežot vajadzīgo elastību.

No otras puses, ASP.NET Core, kas ir Microsoft izstrādāts atvērtā koda ietvars, piedāvā starpplatformas atbalstu gan Windows, gan Linux vidē, nodrošinot augstu veiktspēju un modernu drošības infrastruktūru. Šis ietvars ir lielisks risinājums mūsdienīgu, mākoņbāzētu lietojumprogrammu izstrādei, taču sākotnēji tas var prasa lielāku laika ieguldījumu un pieredzi Microsoft tehnoloģijās, kas reizēm ierobežo tā pieejamību plašākai izstrādātāju auditorijai.

Ņemot vērā projekta specifiskās prasības pēc stabilas aizmugursistēmas, kas prasa drošu datu lasīšanu, rakstīšanu, rediģēšanu un dzēšanu, kā arī efektīvu integrāciju ar MySQL datubāzi, Spring Boot ir izvirzīts kā vispiemērotākais risinājums. Ar savu stabilitāti, drošības funkcijām un mērogojamību Spring Boot nodrošina uzņēmuma līmeņa platformu, kas spēj apmierināt gan pašreizējās, gan nākotnes prasības, kā arī ir aktīva un plašai kopienai atbalsta ietvars. Tādēļ, izvēloties Spring Boot, tiek nodrošināta ne tikai ātra un efektīva izstrāde, bet arī drošība un uzticamība, kas ir būtiski, attīstot liela mēroga komerciālas lietojumprogrammas.

Sistēmas saskarnei ar lietotāju veidosim mājaslapu, un lai to viegli un droši savientu ar izvēlēto aizmugursistēmu izmantošu kādu no ietvaru, kur no izplatītākie ir Angular (TypeScript/JavaScript), React (bibliotēka, ko izstrādājusi Facebook), Vue.js (progresīvs JavaScript ietvars) un Svelte (jauna, kompilatora balstīta pieeja). Katrai no šīm tehnoloģijām ir savi unikālie spēcīgie un vājie punkti, kas ietekmē to piemērotību specifiskiem projektiem.

Angular, izstrādāts un uzturēts Google, piedāvā pilnvērtīgu un struktūrizētu platformu, kas ietver visu nepieciešamo – no iebūvētas datu saišu saistības un atkarību injekcijas līdz veidojamām komponentēm un spēcīgu moduļu sistēmu. Šāds pilnīgs risinājums nodrošina konsekventu izstrādes vidi, kas ir īpaši vērtīga lielāku, kompleksu lietojumprogrammu gadījumos, kur nepieciešama lielāka struktūras un koda uzturēšanas konsekvence. Angular ir paredzēts iespaidīgiem, mērogojamiem projektiem, kur komponentu bagātība un izstrādātāju prakses standartizācija ir būtiska, taču reizēm šī „vissvienā“ pieeja var radīt nevajadzīgu sarežģītību mazākiem projektiem, kā arī lielāku iepazīšanās laiku jaunajiem izstrādātājiem.

React piedāvā vieglu un elastīgu pieeju, ļaujot veidot dinamiskas lietotāja saskarnes ar virsroku uz kompozicionālo arhitektūru un virtuālo DOM manipulācijām, nodrošinot izcilu veiktspēju. Tomēr React kā bibliotēka, nevis pilnīgs ietvars, prasa papildus risinājumus, lai izveidotu pilnvērtīgu projektstruktūru, kas var radīt sarežģījumus liela mēroga un visaptverošu projektu uzturēšanā. Savukārt, Vue.js piedāvā vienkāršu mācīšanās līkni un elastību, kas ļauj ātri uzsākt izstrādi, taču tas var nebūt pietiekami standarta liela mēroga komerciāliem projektiem, kur tiek prasīta stingrāka arhitektūra un ilgstoša uzturēšana. Svelte, kā jauna pieeja, koncentrējas uz koda samazināšanu un augstu veiktspēju, bet tā popularitāte un kopienas atbalsts vēl nav sasniedzis tādu līmeni, kā citiem minētajiem risinājumiem.

Ņemot vērā projekta specifiskās prasības pēc stabilas, drošas un uzturamas frontend sistēmas, kurā ir jānodrošina konsekventa lietotāja saskarne, bagātīga funkcionalitāte un spēja strādāt ar lieliem lietotāju datu apjomiem, Angular izceļas kā vispiemērotākais risinājums. Angular nodrošina pilnīgu struktūru ar skaidri definētiem standartiem un labu integrāciju ar citiem rīkiem un backend sistēmām, kas atvieglo lielāku projektu mērogošanu un uzturēšanu ilgtermiņā. Tādēļ, izvēloties Angular, projektā tiek iegūta droša, konsekventa un spēcīga frontend saskarne, kas ir ne tikai veidota, lai nodrošinātu izcilu veiktspēju, bet arī piemērota pastāvīgai attīstībai un uzturēšanai.

Šis analītiskais apskats sniedz padziļinātu skatījumu uz populārākajiem frontend risinājumiem, integrējot to pozitīvos un negatīvos aspektus vienotā tekstā, un pamatojami par Angular izvēli kā gala risinājumu, kas spēs nodrošināt mūsdienīgas, drošas un mērogojamas lietotāja saskarnes izstrādi.