LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

Kvalitātes nodrošināšanas moduļa izveide uzņēmumā "Meditec"

BAKALAURA DARBS

Autors: **Inga Kauliņa**

Studenta apliecības Nr.: ik14070

Darba vadītājs: Dr.habil.dat., CISA, profesors , LU DF Juris Borzovs

RĪGA 2018

ANOTĀCIJA

Autore, strādājot uzņēmumā “Meditec”, bija saskārusies ar nepieciešamību izveidot kvalitātes nodrošināšanas moduli, ar kura palīdzību lietotāji ar attiecīgajām tiesībām spētu definēt un pārizmantot validācijas kritērijus ievadlaukiem.

Bakalaura darba mērķis bija izpētīt informāciju avotus par vairākām validācijas metodēm un to realizācijām, izpētīt izmantotās tehnoloģijas un realizētās validācijas metodes uzņēmuma “Meditec” veidotajā satvarā, kā arī iegūto informāciju pielietot, lai izveidotu kvalitātes nodrošināšanas moduļa ideju uzņēmumam.

Galvenie uzdevumi ir izpētīt literatūras avotos iegūstamo informāciju par validācijas metožu realizācijām, izpētīt uzņēmuma satvarā realizētās validācijas metodes un izmantotās tehnoloģijas, izstrādāt ideju kvalitātes nodrošināšanas moduļa realizācijai.

Rezultātā ir izpētīti iegūtie materiāli par validācijas metodēm un izveidota konceptuāla ideja kvalitātes nodrošināšanas moduļa realizācijai.

Atslēgvārdi: **kvalitātes nodrošināšanas modulis, ievaddatu validācija, validācijas metodes, modulis, lietotāja ievades validācija.**

ABSTRACT

Creation of quality assurance module in company “Meditec”

The author, working for Meditec, has faced the need to establish a quality assurance module to help users with relevant rights be able to define and use validation criteria for entrypoints.

The purpose of the Bachelor's paper was to study information sources on several validation methods and their realization, to study the technologies used and implemented validation methods in the framework of the company Meditec, as well as to apply the obtained information in order to establish the concept of the quality assurance module for the company.

The main tasks are to study the sources of literature, the realization of validation methods, to study the validations and technologies used in the company framework, to develop an idea for implementation of the quality assurance module.

As a result, the obtained materials about validation methods are investigated and a conceptual idea for implementation of the quality assurance module is created.

Keywords: **quality assurance module, input validation, validation methods, module, user input validation.**

SATURS

APZĪMĒJUMU SARAKSTS

Būla tips - tips, kas raksturo loģiskās (patiess, aplams) vērtības.

GUI – grafiskais lietotāja interfeiss.

HTML - vienkārša iezīmēšanas valoda, ko izmanto, lai izveidotu hiperteksta dokumentus, kas ir pārvietojami no vienas platformas uz citu platformu.

Kontrolieris – Nodrošina metodes, kuras atbild uz HTTP pieprasījumiem, kuri ir nosūtīti uz ASP.NET MVC tīmekļa vietni.

MSDN - Microsoft izstrādātāju tīkls.

MVC-Projektēšanas šablons „Modelis-Skats-Kontrolieris", sadala sistēmu trijās daļās - datu ievads, datu apstrāde, datu attēlojums, kur katra daļa risina savu uzdevumu.

Patents - dokuments, kuru kompetenta valsts institūcija izsniedz izgudrotājam vai viņa tiesību mantiniekam, apliecinot autortiesības.

Trigeris – Mehānisms ar kuru sistēma vai lietotne aktivizē vai nu tūlītēju vai animētu izmaiņu vienā vai vairākās īpašībās.

URL - tīmekļa addrese.

Validācija - datu atbilstības pārbaude konkrētiem, iepriekš noteiktiem kritērijiem.

.NET satvars – Microsoft izstrādāts daudzu klašu bibliotēku kopums, kurš atbalsta vairākas programmēšanas valodas, kā piemēram C# un VisualBasic.

IEVADS

Lietotnes pareizas funkcionalitātes nodrošināšanai ir svarīgi, lai lietotāju ievadītie dati atbilstu konkrētiem kritērijiem. Datu savlaicīga validācija aiztaupa kļūdas sistēmā pie datu saglabāšanas, ielasīšanas, un rediģēšanas nodrošinot lietotājam kvalitatīvāku sistēmas darbību. Apstākļos, kad lietotnes formās validācijas kritērijiem ir tendence mainīties biežāk, ir svarīgi, lai lietotni var ērti un ar salīdzinoši maz resursiem pielāgot pie šīm jaunajām prasībām arī lietotāji.

Šī tēma tika izvēlēta, lai uzņēmuma "Meditec" ietvaros risinātu problēmu par nepieciešamu vienotu ievaddatu validāciju, kuras ietvaros lietotāji varētu definēt sev nepieciešamās validācijas kritērijus konkrētiem laukiem bez izstrādātāju iesaistīšanās procesā.

Veicot materiālu meklēšanu izmantojot "Google scholar" un norādītās atslēgfrāzes "kvalitātes nodrošināšanas modulis", "ievaddatu validācija", "validācijas metodes", "lietotāja ievades validācija" autore atrada tikai vienu patentu apstiprinātu 2003.gadā US6535883B1 un salīdzinoši maz citu materiālu, kuros būtu pieminēta līdzīga metožu veidošana. Meklējot materiālus Latvijas Universitātes noslēguma darbu bibliotēkā tika atrasts tikai viens darbs ar līdzīgu tematu, secinot, ka šī tēma ir maz pētīta zinātniskā līmenī. Tālākajā meteriālu meklēšanā un izpētē autore koncentrējās uz realizācijas metožu izpēti, izmantojot apmācību materiālus Microsoft sertifikātu iegūšanai, kā arī MSDN internetā pieejamos materiālus, meklējot informāciju saistībā ar validācijas metožu izveidošanu.

Šī darba mērķis ir izpētīt vairākas plaši zināmas datu validācijas metodes, kā arī uzņēmumā "Meditec" izmantotās datu validācijas metodes un izveidot risinājumu vienotam datu validācijas modulim.

Pirmajā nodaļā ir aprakstīta datu validācijas nozīmība sistēmā. Izpētot darba vietā pieejamos aktuālos materiālus apmācībām Microsoft sertifikātu iegūšanai, autore uzzināja nozīmīgu informāciju par datu validācijas nozīmību, kā arī informāciju par vairākām validācijas metodēm.

Otrajā nodaļā, balstoties uz izpētītajiem materiāliem pirmajā nodaļā, vietnē "Google scholar" atrasto patentu, kā arī Mikrosoft iztrādātāju tīklā atrastajiem noderīgajiem aprakstiem, autore izpētīja vairākas realizācijas pieejas validācijai, satvara ietvaros un ārpus tiem.

Trešajā nodaļā autore aprakstīja izmantotās tehnoloģijas uzņēmumā “Meditec”, kā arī izpētīja sīkākus piemērus par satvarā realizētajām validācijas metodēm.

Ceturtajā nodaļā, balstoties uz ievākto informāciju, kopā ar sistēmanalītiķi un sistēmas arhitektu autore izstrādāja ideju modulim ar vienotām validācijas metodēm, kur lietotāji ar attiecīgām tiesībām spētu noteikt validācijas kritērijus aizpildāmajiem laukiem sistēmā.

1. DATU VALIDĀCIJAS NOZĪME SISTĒMĀ

1.1. Kas ir datu validācija

LZA TK ITTEA terminu datubāzē datu validācija tiek izskaidrota, kā, "Datora atmiņā vai dokumentos glabājamo datu atbilstības pakāpes reālajam atspoguļojamo objektu stāvoklim pārbaude. Datu validācija nodrošina pārbaudāmo datu atbilstību noteiktajām specifikācijām un izslēdz neparedzētu rakstzīmju un datu tipu izmantošanu vai novirzes no uzdotajiem lauku garumiem" [5]. Datu validācija nodrošina kvalitāti lietotāja ievadītajiem datiem, veicot pārbaudi ar attiecīgiem validācijas noteikumiem, pārbaudot datu pareizību un saturību.

1.2. Datu validācijas nepieciešamības iemesli

"Kad jūsu lietotne ir produkcijā, tai ir jāsaskarās ar dažādiem ievadīšanas tipiem. Daļa no ievades nāk no citām sistēmām, kuras ir integrētas šajā sistēmā, un lielākā daļa no ievades ir lietotāju radīta" [6]. Lietotājus, kuri veic ievadi sistēmā, pēc dotā avota ievada divās grupās:

- **Nevainīgie lietotāji** (*Innocent users*) – "lietotāji, kuri aplikāciju izmanto, lai veiktu darbu un var pieļaut kļūdas veicot darbu lietotne" [6].
- **Ļaunprātīgie lietotāji** (*Malicious users*) – "lietotāji, kuru mērķis ir atrast lietotnē vājos punktus un tos ekspluatēt sava labuma gūšanai" [6].

Datu validācija ir nozīmīga citu sistēmu integrēšanā. Datiem, kuri tiek pārsūtīti starp sistēmām, jābūt atbilstošiem abu sistēmu standartiem, kā arī jābūt attiecīgos formātos, lai sistēmas šos datus savstarpēji varētu pārsūtīt, kā arī pārveidot attiecīgajos formātos, lai dati būtu saprotami [6].

Satvaru sniegtās iespējas palīdz izstrādāt efektīvas validācijas, bet ar tām ne vienmēr ir pietiekoši, lai spētu pilnībā minimizēt kļūdu iespējamību datos, līdz ar to papildus validācijas ārpus satvara robežām palīdz nodrošināt augstāku datu kvalitāti.

1.3. Datu integritātes pārvaldīšana

"Lietotnē nonākuši neadekvāti vai bojāti dati nereti var izraisīt sistēmas sabrukumu, kas izraisa neērtības lietotājiem, bet galvenais ir neļaut šāda veida datu nonākšanu līdz datubāzei, kur

tie var ietekmēt sistēmas integritāti" [6]. Ja sistēmā netiek saglabāti precīzi dati, tas var rezultēties ar nepilnīgu informāciju vai bez svarīgām sasaistēm informācijas sasaistei lielākā kopumā. Lai izvairītos no šāda veida datiem sistēmā, vai aizkavētu vēlamu datu pazušanu šādu iemeslu dēļ, ir nepieciešama datu integritātes pārvaldīšana.

Var iedalīt 4 dažādus tipus datu integritātei:

- **Entitātes integritāte** (*entity integrity*) - "[...] katram ierakstam datubāzē vajadzētu būt unikāli identificētam. Datubāzē tas ir panākts izmantojot primārās atslēgas kolonnu. Primārā atslēga unikāli identificē katru datu rindu. Tā var būs ģenerēta no datubāzes vai no lietotnes" [6].
- **Domēna integritāte** (*domain integrity*) - atsaucās uz ticamību datiem kuri atrodas entītijā. Tas varētu būt par tipu vai iespējamajām vērtībām, kuras ir atļautas [6].
- **Attiecinošā integritāte** (*referential integrity*) - entītiju savstarpējās attiecības [6].
- **Lietotāja definēta integritāte** (*user-defined integrity*) - ietver sevī darījuma noteikumus, kuriem vajag sekot [6].

Datubāžu sistēmās šīs integritātes pārbaudes var atrast jau iestrādātas. Tās savlaicīgi veic pārbaudi, vai visi nepieciešamie savienojumi ir veikti un dati ir sasaistīti akurāti, atgriežot lietotājam paziņojumus par kļūdām vai nesakritībām, atkarībā no datubāzes konfigurācijām.

1.4. Ievadīto datu validācijas process

Datu validācijas procesa uzbūve ir ar loģisku secību:

1. Lietotājs veic datu ievadīšanu.
2. Dati tiek salīdzināti ar iepriekš definētiem validācijas kritērijiem.
3. Ja dati atbilst kritērijiem, tie tiek nodoti tālākai izmantošanai, bet par datiem, kuri nav izturējuši validācijas kritērijus, lietotājam tiek sniegta kāda zīme vai paziņojums.
4. Lietotājam kļūdainu datu gadījumā tiek sniegta iespēja datus kādā veidā labot.

Šāds datu validācijas process ir sastopams visās sistēmās, kurām ir svarīga datu atbilstība konkrētiem iepriekš noteiktiem datu kritērijiem.

2. DATU VALIDĀCIJAS METODES

Sistēmā ievadītajiem datiem, atkarībā no sistēmas uzbūves, varētu būt vairāki validācijas līmeņi. No sākuma ir jāpiemin fakts, ka datu validācija norit vēl pirms datu ievadīšanas sistēmā, atkarībā no personas izvērtēšanas. Tipiskākai sistēmai validācija var notikt datu ievadīšanas laikā lietotāju saskarnē, sistēmas klienta pusē vai servera pusē.

2.1. Datu validācija pirms datu ievadīšanas

Datu validācija sākotnēji norit vēl pirms datu ievades sistēmā. Dati tiek sagatavoti pirms tie tiek ievadīti sistēmā, kur persona atbildīga par datu ievadīšanu pēc savām spriešanas spējām nosaka datu atbilstību nepieciešamajam formātam un kontekstam, kā arī nosaka vai sniegtie dati ir pilnīgi.

2.2. Datu validācija saskarnē

Datu validācija lietotāju saskarnē arī minimālā daudzumā, nosakot, piemēram, lauka tipu vai atļauto simbolu skaitu, jau savlaicīgi var novērst vairākas potenciālās nepilnības datos, nodrošinot veiksmīgu datu konvertāciju, saglabāšanu un apstrādi. Tā kā validācija šajā slānī neietver visu sistēmu, validācija norit ar minimāliem patērētajiem resursiem. Kaut gan validācijas nolūkos laukiem var piemērot daudzus kritērijus sākotnējā lietotāju ievadē, jāatceras, ka šādā veidā pārāk ierobežojot lauka ievades kritērijus, lietotne var kļūt tās lietotājiem daudz neērtāka, tāpēc šāda veida ierobežojumus vēlams veikt gudri un intuitīvi saprotami. Piemēram, ievadot personas kodu, ja ir svarīgi, lai datos būtu attiecīgais noformējums personas kodam, nevajadzētu lietotājam paziņot, ka informācija ir nepareiza ja ievadīta bez starpsvītras, bet laukā jau ievades laikā veikt konvertāciju uz attiecīgo noformējumu.

2.2.1. Veidlapu validācija

"Tiklīdz ir nepieciešams ievākt lietotāju datus, jūs atverat durvis uz savu aplikāciju, tāpēc jums ir jāpārliecinās, ka lietotāju ievade ir droša" [7]. Sistēmu un lietotņu uzbūvē būtu jābūt paredzētiem risinājumiem, kuri nodrošinātu, ka datu ievades laikā, dati sasniedz paredzēto sistēmu un nenonākt neparedzētās vietās.

Lietotāju datu ievadei formās ir svarīgi, lai forma būtu intuitīva un saprotama aizpildīšanai. Parādot kļūdu paziņojumus ar ieteikumiem datu kvalitātes uzlabošanai, pareizajai noformēšanai vai nepieciešamībai obligāti norādīt kādus datus, lietotājiem tas var šķist kaitinoši, kaut gan sistēmas pareizai darbībai tas ir nepieciešami [7].

Nereti ar validāciju formā nepietiek, lai nodrošinātu nepieciešamo datu kvalitāti."[…] ja ir nodrošināta validācija pārlūkā, jums ir vajadzīga arī validācija serverī. Problēma ar validāciju pārlūkā ir tāda, ka validāciju pārlūkā ir viegli apiet uzbrucējiem" [7].

2.2.1.1. Obligātā validācija

"Laikam visvienkāršākais tips veidlapu validācijai ir obligātais (*required*) atribūts uz lielāko daļu no iesniedzamajiem elementiem. Obligātais elements ir Būla tipa atribūts, kas pieprasa lietotājam ievadīt datus pirms nosūtīšanas uz serveri" [7]. Laukā var norādīt viettura (*placeholder*) elementu, kas padarīs aizpildīšanu intuitīvāku, kā arī garantēs aizpildītu lauku, kamēr lietotājs nebūs neveicis lauka aizpildīšanu.

```
<form id="myForm">
    Favorite Car:
    <select name="favoriteCar" required="required">
        <option>Ford Fiesta</option>
        <option value="Chevy">Chevrolet</option>
        <option>BMW</option>
    </select><br />
    Comment:
    <input type="text" name="comment" required="required"/><br />
    Email:
    <input type="email" name="email" required="required"/><br />
    <button type="submit" name="submit">Submit</button>
</form>
```

2.1. att. **Obligātā elementa realizācijas kodā [7].**

2.1. attēlā var aplūkot piemēru, kā obligātais elements ir realizēts kodā. Šiem elementiem varētu ērti pievienot arī vietturus. Šāda veida validācija, kā redzams piemērā, ir ļoti vienkārši realizējama un kopā ar datu validāciju pie informācijas saglabāšanas, kā arī ar konkrētu realizāciju validācijas kritēriju neizturējušo lauku attēlošanai, var sniegt redzamus paziņojumus un orientējošus apzīmējumus, lai paziņotu lietotājam par informācijas trūkumu šajos obligātajos laukos.

2.2.1.2. URL ievades validācija

URL validē pievienojot obligāto elementu ar vietturi pēc izvēles. Šo validāciju var papildināt ar norādītu šablonu, kā nepieciešamajai saitei vajadzētu būt noformētai [7].

```
<form id="myForm">
  website:
  <input type="url" name="website"
    required="required" pattern="https?://.+" />
    <button type="submit" name="submit">Submit</button>
</form>
```

2.2. att. **URL ievades validācijas realizācija kodā [7].**

2.2. attēlā aplūkotajā piemērā, var secināt, ka arī šī validācija ir ērti realizējama kodā savlaicīgi konvertējot ievadītos datus par tīmekļa vietnes adresi.

2.2.1.3. Numuru un diapazonu validācija

Ciparu vērtībām var izmantot elementus ar tipiem numurs *(number)* un diapozons *(range),* kuri pieņem atribūtus mazākais *(min),* lielākais *(max)* kā arī solis *(step)* [7].

```
<form id="myForm">
    Current Age:
    <input type="number" name="age"
        min="18" max="99" value="30"
        required="required" /><br />
    Rating:
    <input type="range" name="rating"
        min="1" max="7" value="4" /><br />
    <button type="submit" name="submit">Submit</button>
</form>
```

2.3. att. **Numuru un diapazonu validācija kodā [7]**

Attēlā 2.3. var aplūkot validācijas realizāciju laukiem age un rating, sniedzot laukam age konkrētu diapazonu (no 18 līdz 99), kā arī sākotnējo vērtību 30 un laukam rating diapazonu (no 1 līdz 7) ar sākotnējo vērtību 4.

2.2.2. Noprotamā (*implicit*) validācija

Noprotamās validācijas gadījumā dati tiek validēti, nolasot katra taustiņa ievadīto informāciju pēc tā nospiešanas, vai arī pēc fokusa pazušanas konkrētajā vadīklā [4]. Tākā šī validācijas metode pieprasa vairāk datora resursu, to plašāk izmanto gadījumos, kad ir pieejami

šādi resursi, lai veiktu lielo skaitu ar atsvaidzināšanām (*refresh*) un rezultāta noteikšanām, kaut gan šajā gadījumā ir momentāli rezultāti un iespēja datus labot, pirms dati ir nosūtīti un datu bāzi.

2.2.3. Izklāstītā (*explicit*) validācija

Izklāstāmo validāciju izmanto kā opciju noprotamās validācijas aizstāšanai. Izklāstītās validācijas gadījumā dati tiek validēti visi uzreiz, pēc pogas "Saglabāt" vai "Nākamais" nospiešanas [4]. Šī metode patērē mazāk datora resursu, jo nav nepieciešama tik bieža atsvaidzināšana un pareizības pārbaude. Metode nodrošina datu pārbaudi tikai pēc visu datu aizpildīšanas un saglabāšanas.

2.3. Klienta puses validācija

Informācijai, kura nonāk lietotnē, jābūt pārbaudītai un apstiprinātai saskaņā ar lietotnes darījuma loģiku. Apstiprinātajiem datiem būtu jābūt attēlotiem lietotāju interfeisā pareizi saskaņā ar lietotnes darījuma loģiku. Lai gan datu validācija var būt veikta lietotnes modulī (*model*), kad serveris saņem informāciju, validācijas prasības var būt padotas no klienta puses, caur skatu (*view*), samazinot vajadzību pēc papildus datiem un aiztures (*lag*). Pēc tam, kad informācija nonāk atpakaļ serverī, tai tiek veikta pēdējā validācija un modelis ir pieejams kontrolierī (*controller*) apstrādei kāda ir tālāk nepieciešama [8].

2.3.1. Modelī (*model*)

"MVC (*model – view - controller*) tipa lietotnē, modelis pārvalda uzvedību un datus lietotnes domēnā" [8]. Modelis validē laukus, kuri nepieciešami MVC tīmekļa lietojumprogrammā, tas nozīmē, kad dati ir validēti pirms saturs ir saglabāts datubāzē [8]. Validācija modelī notiek izmantojot datu anotācijas (*data annotation*) metodi, kur katram laukam nosaka vai tas ir obligāts, kādam datu tipam tajā ir jābūt, kā arī minimālo un maksimālo garumu [8].

```
namespace ArticleApp.Models {
    public class Article {
        public int ID { get; set; }
        [Required]
        [StringLength(50,MinimumLength=5)]
        public string Title { get; set; }
        [RegularExpression[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}")]
        AuthorEmail { get; set;}
        [DataType(DataType.Date)]
        [Range(300, 3000)]
```

2.4. att. **Klientu puses validācijas sadaļa modelī [8]**

2.4. attēlā var aplūkot piemēru datu anotācijas metodei. Izmantojot norādītos kritērijus, tiek nodrošināta loģiski pareiza klases Article objekta aizpildīšana.

2.3.2. Skatā (*view*)

Skats pārvalda informācijas attēlošanu MVC tīmekļa lietojumprogrammā. Uz skatu nonāk jau modelī laukiem piesaistītie validācijas kritēriji. Modeļa balstītas datu anotācijas metode nodrošina skatu, no kura tālāk var veidot klienta puses datu validāciju, kurā tiek nodrošināta ērta lietotāju pieredze, kā arī paātrināta darbība. Validācija servera pusē ir kā nepieciešamība, lai aizkavētu sliktas kavlitātes datu nokļūšanu sistēmā [8].

2.3.3. Kontrolierī (*controller*)

Izmantojot uz modeļa balstītas datu anotācijas principu, jāveido pārbaude uz servera pusi [8]. Īpašībām (*properties*) kontroliera slānī ir iespējams izmantot modelim specifiskus validācijas palīgus kā, piemēram, IsValid, kas sniedz sarakstu ar derīgiem un nederīgiem laukiem, kurus lietotnē var izmantot.

```
[HttpPost]
public ActionResult Create(Article article)
{
    if (ModelState.IsValid)
    {
        db.Articles.Add(article);
        db.SaveChanges();
        return RedirectToAction("Index");
   }

   return View(article);
}
```

2.5. att. **Klientu puses validācijas sadaļa kontrolierī [8]**

2.5. attēlā var aplūkot piemēru klienta puses validācijas sadaļai kontrolierī. Šajā gadījumā IsValid tiek izmantots, lai noskaidrotu ModelState statusu, lai varētu sekmīgi pievienot Article klases objektu article Article tipa objektu sarakstam.

2.3.4. Attālinātā validācija

Šāda validācijas metode ir vairāk nepieciešama, lai lietotnes darbības laikā lietotāji spētu, jau ievadot datus, redzēt to atbilstību validācijas kritērijiem. Piemērs varētu būt sadaļa lietotāju reģistrēšanai lietotnē, lai lietotāji spētu savlaicīgi izvēlēties unikālu lietotājvārdu un varētu pārbaudīt citu datu atbilstību kritērijiem, kādi ir noteikti jaunajiem lietotājiem. Šādā gadījumā ir nepieciešams attālināts validators, kurš noskaidro datu pareizību un atgriež rezultātu [8].

```
public JsonResult IsUserAvailable(string username)
{

    if (!WebSecurity.UserExists(username))
    {
        return Json(true, JsonRequestBehavior.AllowGet);
    }

    string suggestedUID = String.Format(CultureInfo.InvariantCulture,
            "{0} is not available.", username);

    for (int i = 1; i < 100; i++)
    {
        string altCandidate = username + i.ToString();
        if (!WebSecurity.UserExists(altCandidate))
        {
            suggestedUID = String.Format(CultureInfo.InvariantCulture,
           "{0} is not available. Try {1}.", username, altCandidate);
            break;
        }
    }
    return Json(suggestedUID, JsonRequestBehavior.AllowGet);
}
```

2.6. att. **Funkcija lietotājvārda pieejamības noteikšanai [8]**

2.6. attēlā ir redzama funkcija, kura ir izveidota lietotājvārda pieejamības noteikšanai. Funkcija IsUserAvailable apstiprina lietotājvārdu un pārbauda tā esamību sistēmā. Ja lietotājvārds jau ir izmantots, metode atgriež alternatīvu lietotājvārdam, bet ja lietotājvārds nav izmantots tas ir veiksmīgi izturējis validāciju un lietotājs to var izmantot [8].

2.4. Servera puses validācija

Servera pusē, saglabājot datus, dati tiek ievietoti tabulās. Pie tabulu izveides katrai tabulas kolonnai ir iepriekš noteikts datu tips un pieļaujamais garums. Šādā veidā tiek noteikti konkrēti kritēriji tabulā glabātajiem datiem un neatbilstoši dati šajās tabulās netiks saglabāti. Pie situācijas, kurā dati nevar tikt saglabāti konkrētā tabulā, datubāzēs ir izstrādāti mehānismi, kuri sniegs paziņojumu, par datu neatbilstību šiem kritērijiem, kā arī atcels datu saglabāšanu tabulā. Lai novērstu problēmas, kad sistēmas nosūtītie dati servera pusē netiek saglabāti, ir nepieciešams izveidot pārbaudes un knovertācijas visā datu apstrādes procesā sistēmas ietvaros, lai sūtīšanas laikā dati jau būtu attiecīgi pārveidoti, lai atbilstu servera puses kritērijiem.

2.5. Pieejamie patenti validācijas metodēm

Veicot materiālu meklēšanu izmantojot "Google scholar" par izvēlēto darba tēmu, autore sistēmā atrada tikai vienu patentu, kurā tika aprakstīta līdzīgas problēmas risināšana. Patents US6535883B1 piešķirts 2003. gadā ASV. Patentā aprakstītā metode sevī ietver grafisku lietotāja interfeisu, lauku validācijas kritēriju izveidošanai.

2.5.1. Datu validācija izmantojot koku struktūru

Tiek nodrošināts grafisks interfeiss, lai veidotu validācijas kritēriju kopu. Validācijas kritēriju kopa tiek izveidota interaktīvi, izvēloties laukus un tiem pievienojot validācijas kritērijus. Forma ir attēlota kā koka struktūra, kurā sakne (*root node*) attēlo formas nosaukumu, bet mezgli attēlo laukus, kurus formā aizpildīs. Kritēriji ir pievienoti laukiem ar dialogu kastēm un tiek attēloti kā apkšmezgli tiem mezgliem, kuros ir attēloti lauki, veidojot kopīgu validācijas kodu. Pabeigtā koka struktūra tiek pārveidota validācijas kritēriju kopā [2].

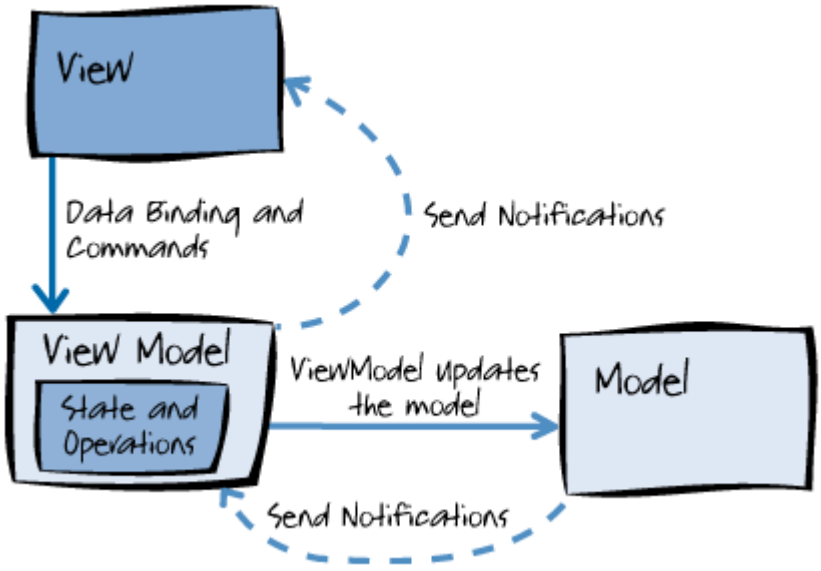
Viena vai vairākas loģikas izteiksmes var būt definētas katram validācijas kritērijam. Katram laukam var būt definēts viens vai vairāki validācijas kritēriji. Izteiksmes ir pievienotas kritērijiem kā apakšmezgli, izmantojot saskarni, izveidotais validācijas koks ar tajā attēlotajiem validācijas kritērijiem tiek saglabāts speciālā validācijas kritēriju failā [2].

3. DATU VALIDĀCIJAS REALIZĀCIJA UZŅĒMUMĀ "MEDITEC"

Uzņēmuma "Meditec" pastāvēšanas laikā ir mainījušies daudzi izstrādātāji, kuri ir strādājuši pie konkrētā projekta. Bez vienota standarta validācijas izveidē, atsevišķiem sistēmas apgabaliem validācija ir izstrādāta citādāk atkarībā no izstrādātāju pieredzes un konkrētajā brīdī nepieciešamo validācijas kritēriju apjoma.

3.1. MVVM arhitektūras paterna pielietojums

Uzņēmuma izstrādātās sistēmas balstās uz vienotu arhitektūras paternu MVVM. MVVM (*Model-View-ViewModel*) var būt lietots uz visām XAML platformām. Tā nodoms ir nodrošināt atdalīšanu problemātiskajām lietām starp lietotāja interfeisa kontroļiem un kontroļu loģiku [9].



3.1. att. **MVVM sistēmas arhitektūras paterns [9]**

Papildus trīs komponenšu pienākumu saprašanai ir ļoti svarīgi arī kā komponentes sastrādājas viena ar otru. Augstākajā līmenī, skats (*view*) "zina par" skata modeli (view model) un skata modelis "zina par" modeli (*model*), bet modelis neko nezina par skata modeli un skata modelis neko nezina par skatu [9]. Skata modelis izolē skatu no modeļa klases un atļauj modelim attīstīties neatkarīgi no skata [9]. Izmantojot šādu arhitektūras paternu, validācijas metodes tiek realizētas ne tikai pie datu ievades servera pusē, bet attiecīgi arī skatā, skata modelī un modelī.

3.2. .NET satvara pielietojums

.NET ir viens no programmatūras satvariem, kuru izstrādājis Microsoft. Tā uzbūvē ietilpst liela izmantojamo satvara klašu bibliotēka, kas sniedz multivalodu sadarbotiesspēju starp vairākām programmēšanas valodām, kuras izmanto .NET satvaru [10]. .NET satvars uzņēmumam sniedz iespējas vieglāk realizēt datu validāciju, piemēram, pielietojot satvara komponenti LINQ (valodas integretētais vaicājums (Language Integrated Query)).

"LINQ paplašina valodu ar vaicājuma (query) izteiksmēm, kas ir līdzīgas SQL paziņojumiem un var būt izmantotas, lai ērti izvilktu un apstrādātu datus no masīviem, enumerētām klasēm, XML dokumentiem, relaciju datubāzēm un trešo personu datu avotiem" [11].

.NET satvars sevī ietver vairākas liederīgas klases validācijas kritēriju veidošanai un datu manipulācijai, lai saglabāšanas momentā dati atbilstu konkrētām validācijas prasībām.

3.1. tabula **Saraksts ar noderīgām .NET satvarā iekļautajām String klases metodēm [1]**

Metode	**Apraksts**
IsNullOrEmpty(String)	"Norāda, vai norādītā virkne ir null vai Tukša virkne." [1].
IsNullOrWhiteSpace(String)	"Norāda, vai norādītā virkne ir null, tukša vai sastāv tikai no atstarpēm."[1].
ToLower()	"Atgriež virknes kopiju konvertētu uz mazajiem burtiem." [1].
Trim()	"Noņem visas atstarpes pirms un pēc pašreizējā virknes objekta" [1].
ToString()	"Atgriež šo instanci no virknes klases" [1].

3.2. tabula **Saraksts ar noderīgām .NET satvarā iekļautajām String klases metodēm [3]**

Metode	**Apraksts**
TryParse(String, Int32)	"Konvertē virknes tipa skaitļa attēlojumu uz tā 32 bitu veseļa skaitļa ar zīmi ekvivalentu. Atgrieztā vērtība norāda vai konvertācija bijusi veiksmīga" [3].
Parse(String)	"Konvertē virknes tipa skaitļa attēlojumu uz tā 32-bitu veseļa skaitļa ar zīmi ekvivalentu" [3].
Equals(Int32)	"Atgriež vērtību, norādot, vai šis gadījums ir vienāds ar norādīto Int32 vērtību" [3].
ToString()	"Pārvērš šī gadījuma skaitlisko vērtību uz tās ekvivalento virknes reprezentāciju" [3].

Aplūkojot tabulas 3.1. un 3.2. var ieraudzīt vairākas lietderīgas un uzņēmumā Meditec arī pielietotas metodes no satvara .NET klasēm.

```
public static string TrimToEmpty(this string value)
{
    if (String.IsNullOrWhiteSpace(value))
    {
        value = String.Empty;
    }
    else
    {
        value = value.Trim();
    }
    return value;
}
```

3.2. att. **Satvara .NET metodes pielietojums**

Attēlā 3.2. var aplūkot piemēru .NET satvara metodes Trim() un IsNullOrWhiteSpace(), izmantošanai datu validācijas metodes realizācijai. Funckija TrimToEmpty() ir izveidota mērķim noņemt slēptas atstarpes pirms vai pēc teksta, lai saglabātie dati nesaturētu atstarpes, kuras varētu ietekmēt sistēmas darbību vai datu apstrādi.

3.3. Validācijas metodes lietotāju saskarnē

Iepazīstoties ar lietotāju saskarnes kodu uzņēmumā "Meditec", autore atrada vairākus piemērus validācijas metodēm lietotāju saskarnē. Visbiežāk validācijas kritēriji izpaužas definētā teksta garumā, vai laukiem noteiktā tipā numuru saglabāšanai.

```
<abx:TextBox
    x:Name="LastNameTextBox"
    AutoTextsEnabled="True"
    Text="{Binding Path=CurrentContact.LastName, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged, Converter={StaticResource TextToFirstCapitalLetterConverter}}"
    MaxLength="100"/>
```

3.3. att. **Saskarnes validācijas metožu lietošanas piemērs**

Aplūkojot attēlu 3.3. var redzēt, ka lauka ievaddati tiek validēti pēc kritērija maksimālajam atļautajam garumam.

3.4. Klienta puses validācijas metodes

Veicot izpēti uzņēmuma "Meditec" veidotajā satvarā un izpētot satvara klienta puses kodu, autore atrada vairākas izstrādē bieži lietotas validāciju metodes.

```
[DataContract]
public class Predicate
{
    [DataMember]
    public int? Id { get; set; }
    [DataMember]
    public string Code { get; set; }
    [DataMember]
    public PredicateScope ScopeType { get; set; }
    [DataMember]
    public int ScopeObjectId { get; set; }
    [DataMember]
    public string OID { get; set; }
    [DataMember]
    public ComparisonOperator ComparisonOperator { get; set; }
    [DataMember]
    public string Value { get; set; }

}
```

3.4. att. **Datu anotācijas metodes lietošanas piemērs**

Viena no pirmajām atrastajām metodēm, aplūkojama attēlā 3.4., ir modelī lietotā datu anotācijas metode. Atbilstoši metodei, šajā piemērā var novērot datiem norādīto tipu un aizpildīšanas nepieciešamību (šajā gadījumā risinātu ar tam paredzēto simbolu pie tipa int).

```
<abx:TextBox
    x:Name="LastNameTextBox"
    AutoTextsEnabled="True"
    Text="{Binding Path=CurrentContact.LastName, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged, Converter={StaticResource TextToFirstCapitalLetterConverter}}"
    MaxLength="100"/>
```

3.5. att. **Attālinātās validācijas metožu lietošanas piemērs**

Vēl viena biežāk lietotā validācijas metode ir attālinātā validācija. Šajā piemērā validāciju aktivizē trigeris PropertyChanged, lai ik pēc katra simbola ievades laukā LastNameTextBox notiktu lauka satura validācija attiecīgi pēc prasībām īpašībai LastName.

```
private void ConfirmNavigationAnswerProcess(DialogBoxAnswerCallbackArgs args, TabNavigationItem tab)
{
    if (!args.Answer.HasValue)
    {
        tab.NavigationConfirmationData.ContinuationCallback(false);
    }
    else if (!args.Answer.Value)
    {
        tab.NavigationConfirmationData.ContinuationCallback(true);
    }
    else
    {
        this._continuationCallback = tab.NavigationConfirmationData.ContinuationCallback;
        this.SaveConfig();
        tab.NavigationConfirmationData.ContinuationCallback(true);
    }
}
```

3.6. att. **Pārveidotāja (*converter*) darbības piemērs**

3.6. attēlā var aplūkot, kā kopā ar validācijas izsaukumu laukā esošā informācija tiek piemērota vēlamajam formatējumam izmantojot pārveidotāju ConfirmNavigationAnswerProcess. Pārveidotājs saņem lauka saturu un atgriež to pārveidojot pirmo burtu par lielo burtu, lai lauka informācija būtu atbilstoša LastName datu vajadzībām.

3.5. Servera puses validācijas metodes

Uzņēmuma aplūkotajā kodā autore atrada arī validācijas metodes servera pusē, kuras galvenokārt balstās uz pārbaudi atbilstošā datu tipa prasībām, simbolu garumam vai lauka, objekta aizpildījumam.

```
IF (NOT EXISTS (
    SELECT
        *
    FROM
        sys.objects
    WHERE
        object_id = OBJECT_ID(N'[QUA].[Function]') AND
        type in (N'U')
))
IF object_id('[QUA].[Function]', 'U') is null
BEGIN
    CREATE TABLE [QUA].[Function](
        [Id]              INT IDENTITY(1,1) NOT NULL,
        [DisplayName]     NVARCHAR(400) NOT NULL,
        [FunctionGroupId] INT NOT NULL,
        [Value]           NVARCHAR(2000) NOT NULL,
        [ErrorMessage]    NVARCHAR(400) NOT NULL
        CONSTRAINT [PK_Function] PRIMARY KEY CLUSTERED ([Id] ASC)
    );
END

GO
```

3.7. att. **Servera puses validācijas metodes piemērs**

4. KVALITĀTES NODROŠINĀŠANAS MODUĻA IZVEIDE

Uzņēmumā “Meditec”, tāpat kā daudzos citos uzņēmumos, datu atbilstība kritērijiem ir ļoti svarīga, līdz ar to sistēmu izstrādē tiek realizētas vairāka veida validācijas metodes gan saskarnes, gan lietotāja puses, gan datubāzes ietvaros. Pagaidām uzņēmuma veidotajos risinājumos validācijas metodes ne vienmēr ir realizētas vienādi un izvēlētie risinājumi validācijām atšķiras uzņēmuma produktos. Vienota un pārizmantojuma moduļa izveide sniedz vienotu risinājumu ievaddatu validācijai, ar potenciālu gan atvieglot izstrādes darbu, gan padarīt sistēmas pielāgošanu lietotājiem pieejamāku.

4.1. Moduļa izveides mērķi un priekšrocības

Kvalitātes nodrošināšanas modulis tiek veidots vairāku tā sniegto priekšrocību dēļ:

- modulis būs pārmantojams citās sistēmās, kuras tiks izstrādātas uzņēmumā “Meditec”, kā daļa no uzņēmuma izstrādāta satvara.
- modulis nodrošinās vienotu validācijas metodi, kā šajā produktā, tā arī nākotnē veidotajos produktos, nodrošinot vienotu principu izstrādātājiem.
- modulis būs ērti un pašsaprotami izmantojams lietotājiem, atvieglojot lauku validāciju pievienošanu bez izstrādātāju iesaistīšanas.
- tiks nodrošināta ātrāka un vienota ievades lauku validācijas kritēriju maiņa atkarībā no pasūtītāja nepieciešamībām.

4.2. Moduļa funkcijas

Galvenās modulim paredzētās funkcijas:

- Intuitīvā veidā lietotājiem ir iespēja definēt validācijas kritērijus konkrētiem laukiem
- Definētu validācijas kritēriju ērta dzēšana vai labošana
- Validācijas kritēriju saglabāšana datubāzē

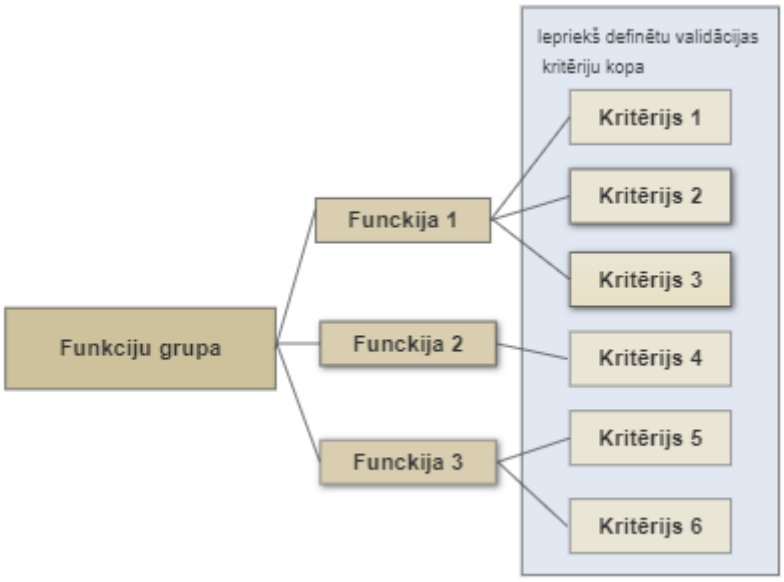
4.3. Moduļa koncepta un realizācijas apraksts

Kopā ar sistēmarhitektu, sistēmanalītiķi un programmētāju, tika izdomāts risinājums, kas sevī iekļautu jau vairākus izstrādē lietotos validācijas paņēmienus, kā arī papildinātu tos ar funkcionalitāti, kas nodrošinātu lietotāju saskarni.

Moduļa pamata funkcionalitāte ir validācijas kritēriju definēšana laukiem. Lai validācijas kritēriju definēšanu katram laukam varētu sistematizēt, validācijas kritēriji tiek veidoti kā izteiksme. No sākuma tiek definēti atsevišķi validācijas kritēriji, kā predikāti, no kuriem tiek sakārtotas validācijas kritēriju izteiksmes.

4.1. att. **Validācijas kritēriju funkciju veidošanas piemērs**

Iepriekš izveidotās validācijas kritēriju izteiksmes ir iespējams aprakstoši nosaukt un saglabāt funkciju grupās, kur tām varēs piekļūt un saglabātās izteiksmes pārizmantot citiem laukiem, attiecīgo lauku norādot ar lauka identifikatora numuru (unikālu katram ievadlaukam). Tā kā izveidotās validācijas kritēriju izteiksmes ir saglabātas izteiksmju grupās, šīs izteiksmes ir iespējams aplūkot un izmantojot lietotāju saskarni var nodrošināt definēto izteiksmju labošanu vai dzēšanu.



4.2. att. **Funkciju grupas uzbūves shēma**

Atskatoties uz otrās nodaļas sadaļu 2.5.1. un validācijas metodi, izmantojot koku struktūru, kvalitātes nodrošināšanas moduļa idejiskais risinājums ir ar līdzīgu principu - izmantojot koka metodi. Patentā minētā koka metode atsaucās uz formas definēšanu kā galveno mezglu, no kura tiek definēti lauki kā koka lapas un lapām ir pievienoti validācijas kritēriji, šajā gadījumā kvalitātes nodrošināšanas modulī par galveno mezglu uzskata katru funkciju grupu, no kuras kā zari tiek pieaudzētas funkcijas, kuras tālāk atzarojas validācijas kritērijos. Uzņēmuma izstrādes principos, katram ievades laukam formā ir savs īpašais kods, kas to padara unikālu. Izmantojot šo unikālo kodu, laukam var piesaistīt izveidoto funkciju grupu. Patentā aplūkotā metode, izveidoto formas koku ar validācijas kritērijiem saglabāja validācijas kritēriju failā, kvalitātes nodrošināšanas moduļa ietvaros validācijas funkciju grupas tiek saglabātas datubāzes tabulās, no kurām tās iegūst pēc nepieciešamības.

Šajā metodē tiks apvienotas vairākas validācijas metodes, kuras tika pieminētas otrajā un trešajā darba nodaļā. Izveidotie validācijas kritēriji, funkcijas un funkcijas grupas glabāsies datubāzē, kur tām būs attiecīgas tabulas. Tabulas būs izveidotas ar kritērijiem, kuri ir paredzēti pareizu validācijas kritēriju definēšanai.

```
IF (NOT EXISTS (
    SELECT
        *
    FROM
        sys.objects
    WHERE
        object_id = OBJECT_ID(N'[QUA].[Predicate]') AND
        type in (N'U')
))
IF object_id('[QUA].[Predicate]', 'U') is null
BEGIN
    CREATE TABLE [QUA].[Predicate](
        [Id]                    INT IDENTITY(1,1) NOT NULL,
        [Code]                  NVARCHAR(50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [ScopeType]             INT NOT NULL,
        [ScopeObjectId]         INT NOT NULL,
        [OID]                   nvarchar(400) NULL,
        [ComparisonOperator]    int NOT NULL,
        [Value]                 nvarchar(max) NULL
        CONSTRAINT [PK_Predicate] PRIMARY KEY CLUSTERED ([Id] ASC)
    );
END
```

4.3. att. **Predikātu tabula datubāzē**

Attēlā 4.3. var aplūkot piemēru vienai no tabulām, šajā gadījumā atsevišķo validācijas kritēriju definēšanai. Redzami ir definēti katras kolonnas formāta kritēriji, kuriem ir jāatbilst datiem, lai tie tiktu saglabāti datubāzē un pareizi apstrādi.

No servera puses dati tālāk nonāk klienta pusē, kur tie tiek apstrādāti pirms izsaukšanas izmantošanai vai saglabāšanai. Dati atrodoties sistēmas apstrādē klienta loģikas sadaļā tiks pārveidoti par validācijas funkciju koku un pielietoti attiecīgā lauka datu validācijai, vai arī tiks saglabāti datu bāzē.

```
[DataContract]
public class Predicate
{
    [DataMember]
    public int? Id { get; set; }
    [DataMember]
    public string Code { get; set; }
    [DataMember]
    public PredicateScope ScopeType { get; set; }
    [DataMember]
    public int ScopeObjectId { get; set; }
    [DataMember]
    public string OID { get; set; }
    [DataMember]
    public ComparisonOperator ComparisonOperator { get; set; }
    [DataMember]
    public string Value { get; set; }

}
```

4.4. att. **Predikāta objekta klase sistēmas klienta pusē**

Attēlā 4.4. var aplūkot vienu no realizētām validācijas metodēm, lai pareizi saņemtu un apstrādātu Predicate objektu, jeb individuālu validācijas kritēriju. Šajā situācijā ir realizēta datu anotācijas metode, kas nosaka datiem vēlamo tipu un aizpildīšanas nepieciešamību.

Saskarnē, kuru var aplūkot 1. pielikumā, arī tiks realizētas vairākas no saskarnes validācijām. Pie pogu "Saglabāt" nospiešanas, tiks veikta noprotamā validācija, lai noskaidrotu vai pirms datu saglabāšanas ir aizpildīti lauki vismaz vienam validācijas kritērijam, vai validācijas kritēriju funkcija sastāv no vismaz viena kritērija utt. Būs nepieciešama arī obligātā validācija, lai noteiktu kuri lauki pie validācijas kritēriju, kritēriju funkciju kā arī funkciju grupu izveides un saglabāšanas būs obligāti aizpildāmi.

Visas izmantotās validācijas metodes un citas, kuras būs nepieciešams pievienot turpmākā moduļa izstrādē, nodrošinās lietotājiem ērtu kvalitātes nodrošināšanas moduļa lietošanu. Tākā modulis ir galvenokārt paredzēts lietot cilvēkiem ar attiecīgām tiesībām, kuri būs atbildīgi par jaunu validācijas kritēriju saglabāšanu sistēmā, tā saskarnes ērta un intuitīva izmantošana būs svarīga. Lietotāju veidotie validācijas kritēriji, funkcijas un funkciju grupas tiks izsauktas lietotnes klienta puses loģikā, apvienojot laukiem iestrādātas validācijas metodes saskarnē un servera pusē, kopā ar lietotāja pievienotajiem validācijas kritērijiem.

REZULTĀTI

Bakalaura darba tēma bija kvalitātes kontroles moduļa izveide uzņēmumā “Meditec”. Balstoties uz bakalaura darbā izvirzīto mērķi izveidot kvalitātes kontroles moduli tika izpēti vairāki materiāli par validācijas realizācijas metodēm un pieejām, tika izpētītas izmantotās tehnoloģijas un realizētās metodes uzņēmuma “Meditec” ietvaros, kā arī izstrādāta ideja kvalitātes nodrošināšanas modulim.

Darba izstrādes laikā autore uzzināja, ka līdzīgu problēmu risināšanai idejas pastāv, taču tās nav plaši aprakstītas un par tām ir maz zinātniskā tipa materiālu. Lai gan par konkrētās problēmas risināšanu materiālu ir maz, ir pieejami daudz materiāli par validācijas realizācijas metodēm kopumā klienta pusē, servera pusē, kā arī saskarnē.

Autore ir apmierināta ar iegūto rezultātu, jo ieguva daudz praktisku zināšanu par validācijas metožu realizāciju, kā arī pēc uzņēmuma vēlmēm izstrādāja ideju kvalitātes kontroles modulim.

Autore veiksmīgi ir uzsākusi kvalitātes kontroles moduļa veidošanu un pagaidām modulis atrodas izstrādes procesā ar turpmākie plāniem moduli attīstīt, nodot testēšanai un vēlāk ieviest arī produkcijā. Pievienotie koda piemēri ir pievienoti saskaņā ar prasībām no darba vietas un tikai atļautie saskaņā ar šīm prasībām.

SECINĀJUMI

Darbs izstrādāts, vadoties pēc uzstādītā mērķa izpētīt izpētīt informāciju avotus par vairākām validācijas metodēm un to realizācijām, izpētīt izmantotās tehnoloģijas un realizētās validācijas metodes uzņēmuma "Meditec" veidotajā satvarā, kā arī iegūto informāciju pielietot, lai izveidotu kvalitātes nodrošināšanas moduļa ideju uzņēmumam. Darba gaitā autorei radās šādi secinājumi:

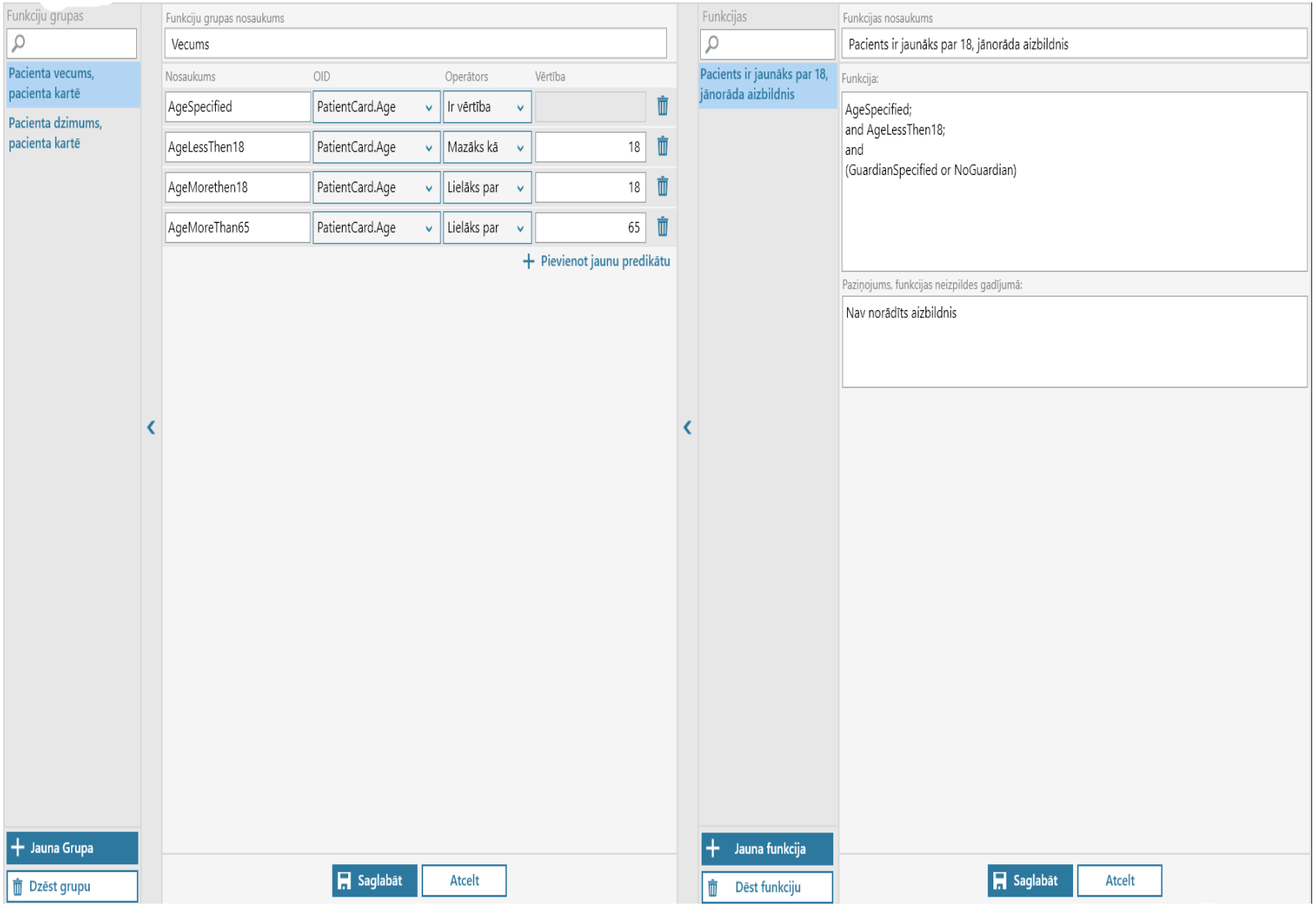
- Ir pieejami daudzi materiāli, par to, kā pareizi realizēt validācijas metodes klienta pusē, servera pusē un lietotāja saskarnē.
- Realizējot uzdevumu, par vienotu veidu, kā validēt laukus, visbiežāk tiek izmantots risinājums ar koka struktūru.
- Veidojot metodi vienotai datu validācijas kritēriju noteikšanai, nevar iztikt bez pašu kritēriju validācijas, lai modulī šos kritērijus varētu izmantot.
- Kvalitātes nodrošināšanas moduļa darbība neaizstās visas validācijas metodes, kuras ir noteiktas sistēmā, bet palīdzēs attiecīgajiem lietotājiem papildināt validācijas kritēriju kopu ievades laukiem.
- Pakļaujot datus vairākiem validācijas posmiem atkarībā no pieejamajiem resursiem sistēmā vai datorā un sistēmas uzbūves, dati apstrādes beigās tiks saglabāti kvalitatīvāki.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. String Methods [Pieejams Internetā]: < https://msdn.microsoft.com/en-us/library/system.string_methods(v=vs.110).aspx> [17.05.2018].
2. System and method for creating validation rules used to confirm input data [Patents, Pieejams Internetā]: < https://patents.google.com/patent/US6535883B1/en > [10.04.2018].
3. Int32 Methods [Pieejams Internetā]: < https://msdn.microsoft.com/en-us/library/system.int32_methods(v=vs.110).aspx> [17.05.2018].
4. User input validation in windows forms [Pieejams Internetā]: < https://msdn.microsoft.com/en-us/library/ms229603(v=vs.110).aspx >.
5. Akadēmiskā terminu datubāze [Pieejams Internetā]: < http://termini.lza.lv/> [10.04.2018].
6. Exam Ref 70-483: Programming in C#. Wouter de Kort. 2013.
7. Training Guide: Programming in HTML5 with JAVAScript and CSS3. Glenn Johnson 2013.
8. Exam Ref 70-486: Developing ASP.NET MVC 4 WEB applications. William Penberthy 2013.
9. The MVVM pattern [Pieejams Internetā]: < https://msdn.microsoft.com/en-us/library/hh848246.aspx> [10.05.2018].
10. .NET framework [Pieejams Internetā]: < https://en.wikipedia.org/wiki/.NET_Framework> [12.05.2018].
11. Language Integrated Query [Pieejams Internetā]: < https://en.wikipedia.org/wiki/Language_Integrated_Query> [17.05.2018].

PIELIKUMS

1.pielikums Saskarnes skice kvalitātes kontroles modulim



Bakalaura darbs “Kvalitātes nodrošināšanas moduļa izveide uzņēmumā “Meditec”” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Inga Kauliņa _____ __.05.2018,

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Darba vadītājs: Dr.habil.dat., CISA, profesors, LU DF Juris Borzovs _____ __.05.2018

Recenzents: Asoc. prof., Dr.sc.comp. Zane Bičevska

Darbs iesniegts Datorikas fakultātē 28.05.2018.

Dekāna pilnvarotā persona:

vecākā metodiķe Ārija Sproģe_____

Darbs aizstāvēts bakalaura darbu gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____.

Komisijas sekretārs: _____