LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

INFORMĀCIJAS SISTĒMU SASKARNES PERSONALIZĀCIJA

BAKALAURA DARBS

Autors: **Kalvis Upītis**

Stud. apl. nr.: ku07001

Darba vadītājs: Dr. dat. Ģirts Karnītis

RĪGA 2011

ANOTĀCIJA

Bakalaura darbā autors apraksta personalizācijas risinājumu ieviešanas nepieciešamību informācijas sistēmās un apskata dažādas iespējas, kā programmatūrā izstrādāt šādu risinājumu.

Darba teorētiskajā daļā ir aprakstīti personalizācijas veidi, personalizācijas ieguvumi un trūkumi, kā arī ir apskatīti dažādi piegājieni, kā informācijas sistēmās ir izstrādāti personalizācijas risinājumi. Darba praktiskajā daļā ir aprakstīts autora realizēts personalizācijas risinājums dokumentu vadības sistēmā FIBU.

Darba mērķis ir izpētīt iemeslus personalizācijas risinājuma izveidei, atrast personalizācijas kritērijus, kā arī izpētīt piemērotāko personalizācijas risinājuma izveides veidu, ko implementēt FIBU programmatūrā.

Atslēgvārdi

Lietotāju saskarnes personalizācija, lietotāju saskarnes aprakstīšanas valodas, FIBU dokumentu vadības sistēma.

ABSTRACT

The Bachelor's paper describes the necessity of personalization solutions in information systems and reviews the different ways how to develop a personalization solution.

The theoretical part describes the types of personalization, the personalization benefits and drawbacks, and looking at various ways how other information systems are designed for personalization solutions. The practical part of the paper describes for document management system FIBU which is implanted by the author.

The goal of the paper is to investigate the causes of personalization solutions development, find personalization criteria, and to find the most appropriate solution for creation of a personalization solution for FIBU software.

Keywords

User interface personalization, user interface markup languages, FIBU document managament system FIBU.

SATURS

APZĪMĒJUMU SARAKSTS

Apzīmējumi		
FIBU	SIA „Datorikas institūts DIVI" izstrādāta datu pārvaldības programmatūra	
Saīsinājumi		
AJAX	Asynchronous JavaScript and XML	Tehnoloģiju kopums dinamiska tīmekļa vietnes satura nolasīšanai un attēlošanai
API	Application programming interface	Specifikāciju kopums, kas ļauj lietotnēm savstarpēji komunicēt, izmantojot to publiskās saskarnes
CSS	Cascading Style Sheets	Valoda tīmekļa vietņu stila aprakstīšanai
GUI	Graphical User Interface	Grafiska lietotāja saskarne
HTML	HyperText Markup Language	Iezīmēšanas valoda informācijas attēlošanai tīmekļa vietnēs
INI	Initialization	Strukturēts konfigurācijas failu formāts
JSON	JavaScript Object Notation	Datu apmaiņas formāts strukturētai datu pārraidei
MVC	Model-view-controller	Programmatūras arhitektūras modelis, kura galvenais princips ir programmatūras loģikas līmeņa nošķiršana no programmatūras skata līmeņa
PHP	Hypertext Preprocessor	Programmēšanas valoda dinamisku tīmekļa vietņu veidošanai
UIML	User Interface Markup Language	Lietotāju saskarnes aprakstīšanas valoda
WPF	Windows Presentation Foundation	Datora-programmatūras grafiskā apakšsistēma darbvirsmas lietojumprogrammu lietotāju saskarņu renderēšanai
WYSIWYG	What you see is what you get	Pieeja, kad informācijas sistēmas saturs rediģēšanas laikā izskatās maksimāli pietuvināts tam izskatam,

		kādā tas tiks attēlots sistēmā
XAML	Extensible Application Markup Language	Microsoft izstrādāta lietotāju saskarnes aprakstīšanas valoda, ko izmanto lietotāju saskarnes aprakstīšanai WPF un *Silverlight* lietotnēm
XML	Extensible Markup Language	Iezīmēšanas valoda strukturētu datu apzīmēšanai

IEVADS

Izstrādājot universālu programmatūras risinājumu, ko iespējams izmantot dažādu nozaru klientu dažādu produktu izveidei, bieži sastopama būtiska problēma ir tas, ka nav iespējams izveidot universālu lietotāju saskarni, kas apmierina visu klientu un visu produktu vajadzības un vēlmes. Lai atrisinātu šādu problēmu, produktā ir jāimplementē personalizācijas iespējas, kas vienā risinājumā ļauj izveidot un neatkarīgi uzglabāt dažādas atšķirīgām prasībām pielāgotas lietotāju saskarnes.

Bakalaura darba mērķis ir izveidot lietotāju saskarnes personalizācijas iespējas FIBU informācijas sistēmā. Lai sasniegtu šo mērķi, darba autors apskata potenciālos personalizācijas iespēju izveides ieguvumus un problēmas, kritērijus, kas jāņem vērā, personalizējot informācijas sistēmas, kā arī veidus, kā ir iespējams realizēt lietotāju saskarnes personalizāciju. Pēc darba autora domām konkrētajam risinājumam vispiemērotākais personalizācijas realizācijas veids – lietotāju saskarnes aprakstīšanas valodu pielietošana – bakalaura darbā tiek apskatīts detalizētāk – tiek pētīta šādu valodu uzbūve un iespējas.

Bakalaura darba struktūra ir šāda:

- darba pirmajā nodaļā ir aprakstīta bakalaura darbā izpētītās problēmas nostādne – iemesli, kāpēc FIBU programmatūrā nepieciešams izveidot lietotāju saskarnes personalizācijas iespējas, kā arī izstrādājamā risinājuma prasības un ierobežojumi;
- darba otrajā nodaļā ir aprakstīti lietotāju saskarnes personalizācijas veidi, personalizācijas ieguvumi un trūkumi, kā arī uzskaitīti personalizācijas kritētiji. Šajā nodaļā ir parādīti konkrēti personalizācijas piemēri citās informācijas sistēmās;
- darba trešajā nodaļā ir izpētītas lietotāju saskarnes aprakstīšanas valodas – veids, kas izvēlēts izstrādājamā personalizācijas risinājuma konfigurācijas uzglabāšanai;
- darba ceturtajā nodaļā ir aprakstīts un demonstrēts darba autora izstrādātais risinājums FIBU programmatūras lietotāju saskarnes personalizācijai – aprakstīta risinājuma vajadzībām izstrādātā lietotāju saskarnes aprakstīšanas valoda, aprakstīta risinājuma uzbūve, kā arī aprakstīts personalizācijas nolūkiem izveidotais lietotāju saskarnes redaktors.

1. PROBLĒMAS NOSTĀDNE

FIBU ir SIA „Datorikas institūts DIVI" izstrādāts datu pārvaldības risinājums, kas nodrošina dažādu veidlapu datu reģistrēšanu, strukturēšanu, apkopošanu, analīzi un plūsmu [1]. FIBU risinājuma izstrādē kā programmētājs ir iesaistīts arī bakalaura darba autors. FIBU informācijas sistēmas arhitektūra ir modulāra un izmantojama dažādu atšķirīgu produktu izstrādei.

Izmantojot FIBU risinājuma kodolu, ir izveidoti dažādi produkti, ko izmanto:

- valsts budžeta un pašvaldības budžetu pārskatu veidošanai,
- finansēšanas plānu un tāmju veidošanai,
- novadu pašvaldību un to padotībā esošo iestāžu lietvedībai,
- dokumentu vadībai,
- dažādu statistikas datu apkopošanai - pārskatiem par gaisa aizsardzību, uzņēmuma serveru struktūras reģistrēšanai, klientu uzskaitei u.c..

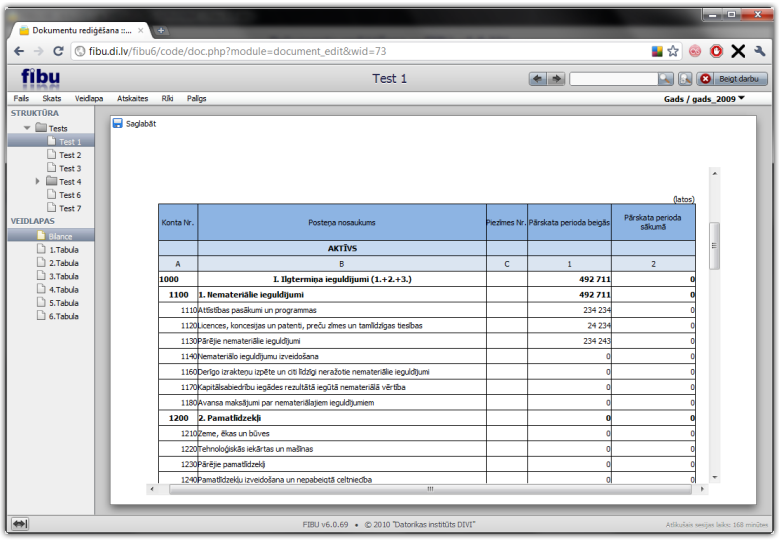
Ņemot vērā plašo FIBU informācijas sistēmas pielietojumu un atšķirīgo klientu loku, ir neiespējami izstrādāt universālu programmatūras lietotāja saskarni, kas atbilst visu klientu specifiskajām vajadzībām un vēlmēm.

Lai FIBU izstrādātājiem paralēli nebūtu jāuztur katram klientam specifiska programmatūras versija ar vienādu biznesa loģikas līmeni, bet atšķirīgu lietotāja saskarnes dizainu, tika nolemts izstrādāt iespēju personalizēt FIBU risinājuma lietotāja saskarni, pielāgojot to katra klienta specifiskajām vajadzībām un vēlmēm. Sākotnēji FIBU lietotāju saskarnes personalizāciju paredzēts veikt tikai tās izstrādātājiem, izveidojot sistēmas lietotāju grupām atšķirīgas lietotāju saskarnes. Ir paredzēts, ka nākotnē daļa no personalizācijas iespējām tiks atļautas arī informācijas sistēmas gala lietotājiem, ļaujot izveidot individuāli pielāgotu lietotāju saskarni.

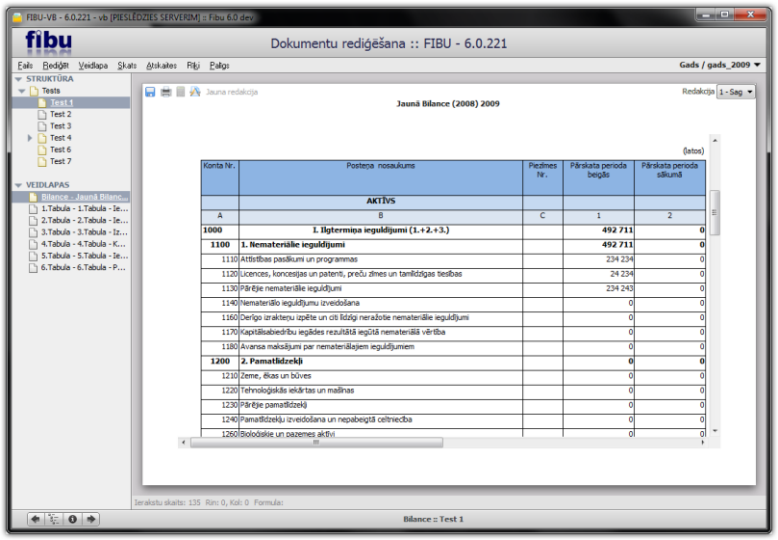
Atkarībā no klienta ir nepieciešams norādīt ne vien specifiskas konkrētas dizaina īpašības (piemēram, saskarnes fona attēlu, elementu izkārtojumu un to iezīmes), bet arī dažādu saskarnē izmantoto vadīklu kopu un dažādu lietotājam veicamo darbību secību (piemēram, atkarībā no tā, vai klients datus grupēs dimensijā „gads", nepieciešams norādīt, vai sistēmā jāiekļauj perioda

izvēlnes vadīkla, kā arī tas, vai, darbu uzsākot, pirms konkrētas veidlapas izvēlnes sākotnēji nepieciešama perioda izvēles darbība).

FIBU informācijas sistēmai ir izstrādāta gan programmatūras tīmekļa versija (1.1. att.), gan programmatūras darbvirsmas versija (1.2. att.).



1.1. att. **FIBU informācijas sistēmas tīmekļa saskarne**



1.2. att. **FIBU informācijas sistēmas darbvirsmas saskarne**

Abas šīs versijas gan pēc izskata, gan funkcionāli ir ļoti līdzīgas (galvenā atšķirība - izmantojot darbvirsmas programmatūru, ir iespējams strādāt arī bez tīmekļa pieslēguma, datus

saglabājot lokālajā datu bāzē). Sākotnēji personalizācijas iespējas ir nepieciešamas tikai tīmekļa programmatūras versijā, bet ir jāparedz iespēja, lai līdzīgi realizētas personalizācijas iespējas ir iespējams izstrādāt arī darbvirsmas programmatūras versijā. Šī iemesla dēļ ir vēlams, lai izstrādātais personalizācijas risinājums būtu platform-neatkarīgs, un vienreiz veicot lietotāju saskarnes personalizāciju, izmaiņas būtu redzamas gan programmatūras tīmekļa versijā, gan arī programmatūras darbvirsmas versijā.

2. LIETOTĀJU SASKARNES PERSONALIZĀCIJA

Personalizācija ir tehnoloģiju kopums, kas ļauj piemērot vienu objektu dažādu lietotāju vai lietotāju grupu individuālajām vajadzībām un vēlmēm. Termins personalizācija tiek lietots dažādās jomās, piemēram, medicīnā, televīzijā, mārketingā. Taču, pārsvarā tas tiek attiecināts uz informācijas tehnoloģijām, īpaši uz tīmekļa tehnoloģijām, kur pēdējos gados dažādu personalizācijas risinājumu ieviešana tiek veikta aizvien biežāk. Lietotāju saskarnes personalizācija ir tehnoloģiju kopums, kas ļauj piemērot informācijas sistēmu lietotāju saskarni lietotāju vai lietotāju grupu individuālajām vajadzībām un vēlmēm, ļaujot dažādiem lietotājiem vienā informācijas sistēmā darboties, izmantojot dažādas lietotāju saskarnes.

2.1. Personalizācijas veidi

Raugoties pēc dažādiem kritērijiem, personalizācijas risinājumus ir iespējams sadalīt vairākās grupās.

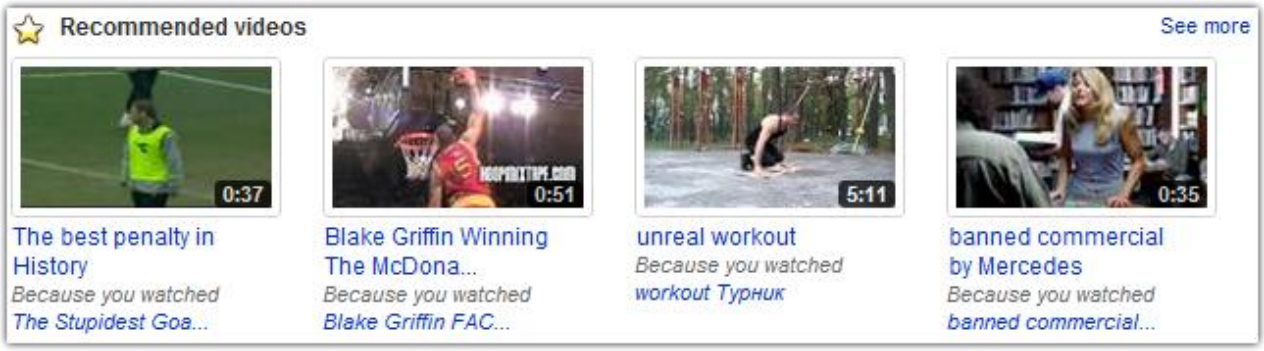
2.1.1. Personalizācijas iedalījums pēc automatizācijas pakāpes

2.1.1.1. Automātiska personalizācija

Šajā grupā tiek iekļauti tie personalizācijas risinājumi, kas tiek veikti automātiski. Automātiska informācijas sistēmas pielāgošana ir iespējama, balstoties uz lietotāja veikto darbību statistiku vai iepriekš ievāktu informāciju par lietotāju. Zinot šādu informāciju ir iespējams prognozēt konkrētā lietotāja ierastās darbības un vēlmes un izcelt lietotāja biežāk izmantotās informācijas sistēmas sastāvdaļas.

Automātiska personalizācija notiek, piemēram, video failu straumēšanas vietnē *YouTube*, kurā, analizējot lietotāja iepriekš skatīto video failu vēsturi, lietotājam tiek ieteikti pēc satura līdzīgi, lietotājam potenciāli interesanti video (2.1. att.). Katrs video fails šajā vietnē tiek ievietots noteiktās kategorijās, kā arī atbilstoši faila saturam tam tiek pievienotas vairākas video raksturojošās birkas (piemēram, *music, 1970s, rock, the beatles*). Uzkrājot statistiku, kādu kategoriju video visbiežāk ir vērojis lietotājs, kā arī kādas birkas šiem video ir pievienotas,

lietotājam tiek piedāvāti video, ko lietotājs vēl nav redzējis un kas ir iekļauti šajās kategorijās un kam ir pievienotas līdzīgas birkas. Tādējādi tiek secināts, ka piedāvātajiem video failiem saturs ir līdzīgs ar lietotāja iepriekš skatītajiem video.



2.1. att. **YouTube automātiski ieteiktie video**

2.1.1.2. Manuāla personalizācija

Šajā grupā tiek iekļauti personalizācijas risinājumi, kuros lietotājs ir tieši iesaistīts informācijas sistēmas pielāgošanā. Izmantojot informācijas sistēmā iekļauto lietotāju saskarnes redaktoru, lietotājs var norādīt kādu informāciju un kādā izskatā viņš vēlas redzēt (pielāgojamās iespējas ir atkarīgas no konkrētās programmatūras specifikas).

Šāds personalizācijas risinājums ir izveidots populārajā sociālajā portālā *Twitter*. Šajā vietnē, izmantojot vietnes iestatījumu paneli, ir iespējams personalizēt lietotāja profila izskatu – norādīt fona attēlu, fona krāsu, tekstu un sāna paneļa krāsas (2.2. att). Šādas iespējas bieži vien izmanto gan uzņēmumi, gan privātpersonas ne tikai, lai savu profilu padarītu vizuāli glītāku, bet arī mārketinga nolūkos, fona attēlā iekļaujot informāciju par viņu darbību.

2.2. att. **Noklusētais (augšā) un personalizēts lietotāja profila izskats vietnē Twitter**

2.1.1.3. Daļēji automātiska personalizācija

Šajā grupā tiek iekļauti personalizācijas risinājumi, kas daļēji atbilst abu iepriekšminēto grupu principiem. Piemēram, lietotājam, balstoties uz informācijas sistēmā uzkrātajiem datiem par viņu, tiek piedāvāts kāds konkrēts personalizācijas variants, ko lietotājs var manuāli akceptēt, noraidīt vai papildus pielāgot savām vēlmēm.

2.1.2. Personalizācijas iedalījums pēc privātuma pakāpes

2.1.2.1. Publiskā personalizācija

Šajā grupā tiek iekļauti personalizācijas risinājumi, kuros lietotāja pielāgotie personalizācijas iestatījumi ir redzama arī citiem informācijas sistēmas lietotājiem. Piemēram, jebkuram vietnes *Twitter* apmeklētājam ir iespējams apskatīt citu lietotāju personalizēto profila izskatu.

2.1.2.2. Privātā personalizācija

Šajā grupā tiek iekļauti personalizācijas risinājumi, kuros lietotāja veiktie konfigurācijas dati tiek slēpti citiem informācijas sistēmas lietotājiem. Ne vienmēr informācijas sistēmas lietotāja personalizētā informācija ir tāda, ko vēlams redzēt citiem lietotājiem, jo ne vienmēr lietotāji ir ieinteresēti publicēt savus iestatījumus, kā arī, iespējams, tā satur sensitīvus datus. Tāpēc dažkārt citiem informācijas sistēmas lietotājiem lietotāja personalizācijas iestatījumus nav vēlams rādīt.

2.1.3. Personalizācijas iedalījums pēc pielāgotā satura veida

2.1.3.1. Satura / saišu personalizācija

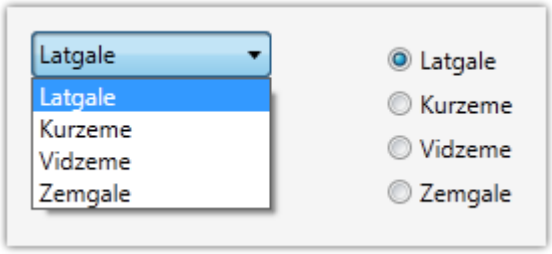
Šajā grupā tiek iekļauti personalizācijas risinājumi, kuros katram lietotājam individuāli tiek pielāgotas informācijas sistēmā redzamais saturs vai saites uz citiem sistēmas resursiem. Šāda personalizācija pārsvarā ir automātiska, balstīta uz iepriekš ievāktu informāciju vai statistiku par lietotāju (skatīt 2.1.1.1. nodaļā apskatīto piemēru).

2.1.3.2. Izskata personalizācija

Šajā grupā tiek iekļauti personalizācijas risinājumi, kuros ir iespējams pielāgot informācijas sistēmas izskatu un dizainu, piemēram, informācijas sistēmas krāsu toņus vai informācijas bloku izkārtojumu. Neatkarīgi no izvēlētā izskata, informācijas sistēmā parādītā informācija paliek nemainīga (skatīt 2.1.1.2. nodaļā apskatīto piemēru).

2.1.3.3. Izmantojamo vadīklu personalizācija

Šajā grupā tiek iekļauti risinājumi, kuros ir iespējams pielāgot veidus, kā lietotājs veic vienu konkrētu uzdevumu. Respektīvi, ja eksistē vairākas vadīklas, ar kuru palīdzību ir iespējams veikt vienu un to pašu uzdevumu (piemēram, no ierobežota iespēju klāsta izvēlēties vienu, iespējams gan, izmantojot izkrītošo izvēlni, gan radio pogas (2.3. att.)), personalizācijas risinājums atļauj lietotājam izvēlēties, kuras vadīklas viņš vēlas izmantot šī uzdevumu paveikšanai.



2.3. att. **Dažādas vadīklas, ar ko iespējams paveikt vienu uzdevumu**

2.2. Ieguvumi no personalizācijas risinājumu ieviešanas

Lai arī lietotāju saskarnes personalizācijas iespēju izstrādāšana lietojumprogrammā prasa papildu resursu izlietojumu un lietotņu izstrādātājiem ne vienmēr tas šķiet nepieciešams resursu pielietojums, šādu iespēju izstrādāšana dod ieguvumus ne tikai informācijas sistēmas lietotājiem, bet arī pašiem izstrādātājiem. Šajā nodaļā ir aprakstīti gan informācijas sistēmu lietotāju, gan izstrādātāju ieguvumi.

2.2.1. Lietotāju ieguvumi

2.2.1.1. Patīkama darba vide

Iespējas informācijas saskarni personalizēt ļauj lietotājam to izveidot attiecīgi lietotāja individuālajai gaumei, kas nozīmē, ka lietojumprogrammas vide būs lietotājam ērtāka un patīkamāka.

2.2.1.2. Ierasta darba vide

Lietotājs, strādājot ar kādu lietojumprogrammu, izmantojot redzes un taustes atmiņu, atceras darbus un kustības, kas jāveic, lai paveiktu noteiktu uzdevumu. Šī iemesla dēļ, ja lietotājam ikdienā ir jāstrādā ar vairākām lietojumprogrammām, īpaši, ja to funkcionalitāte ir līdzīga, ir ērti, ja tās ir savstarpēji līdzīgas. Ja personalizācijas iespējas dod iespēju lietojumprogrammas izskatu padarīt līdzīgu citām lietotāja izmantotajām lietotnēm, tas padara lietotāja darbu ērtāku un ātrāku.

2.2.1.3. Kvalitatīvāka informācija

Iespējas katram lietotājam izvēlēties, kādu saturu viņš vēlēsies skatīties informācijas sistēmā vai arī informācijas automātiska filtrēšana attiecīgi lietotāja interesēm ļauj lietotājam koncentrētāk un pārskatāmāk iegūt tikai viņu interesējošu informāciju, kas konkrētajam lietotājam šķiet kvalitatīvāka, jo pēc iespējas maz satur lietotājam pitenciāli neinteresantu informāciju.

2.2.1.4. Iespējas paveikt darbu ātrāk

Padarot darbu ar informācijas sistēmu lietotājam ērtāku un patīkamāku, kā arī ļaujot lietotājam izmantot iepriekšējo pieredzi, ir iespējams panākt lietotāja veicamo uzdevumu ātrāku izpildi, tādējādi ietaupot lietotāja laika un līdz ar to arī naudas resursus.

2.2.2. Izstrādātāju ieguvumi

2.2.2.1. Iespējas pielāgot vienu sistēmu dažādu klientu vēlmēm

Izstrādātājiem, kuri veido vairākiem pasūtītājiem pielāgojamus produktus, ne reti nākas saskarties ar gadījumiem, kad katra pasūtītāja vēlmes ir krasi atšķirīgas, pat attiecībā uz informācijas sistēmas saskarni. Turklāt, dažādu klientu vēlmes ir savstarpēju pretrunīgas. Ja informācijas sistēma izstrādā iespēju personalizēt sistēmu dažādām lietotāju grupām (šajā gadījumā - dažādiem klientiem), ir iespējams paralēli uzturēt viena produkta vienas versijas dažādas instances ar vienādu biznesa loģikas līmeni, no kurām katra instance ir pielāgota attiecīgā pasūtītāja vēlmēm.

2.2.2.2. Iespējas veikt nelielas izmaiņas sistēmā bez jaunas versijas nosūtīšanas

Atkarībā no izstrādātāju un pasūtītāju iekšējiem noteikumiem, līguma nosacījumiem un dažādiem papildu faktoriem, jaunas programmatūras versijas nodošana ekspluatācijas vidē var būt vairākas dienas vai pat vairākas nedēļas ilgs process. Ja pasūtītāja pieteiktās izmaiņas ir nebūtiskas un tās neietekmē informācijas sistēmas biznesa loģikas līmeni, šāds noilgums nav

nepieciešams. Šādas problēmas atrisinās, ja šīs izmaiņas, ir iespējams veikt, izmantojot sistēmā iestrādātos personalizācijas mehānismus.

2.2.2.3. Iespējas sistēmu integrēt ar citām sistēmām

Mūsdienās aizvien biežāk, veidojot informācijas sistēmas, tajās izmanto gatavas citu izstrādātāju veidotas komponentes vai arī informācijas sistēmā iekļauj citas informācijas sistēmas izsaukumu, kas veic nepieciešamo uzdevumu. Piemēram, dažādu aptaujas formu aizpildīšanai var izmantot *Google Docs* veidlapas [2], bet interaktīvas kartes attēlošanai - *Google Maps* karšu sistēmu [3]. Lai arī šāda citu informācijas sistēmu iekļaušana ir ērta un ātra, jo nav nepieciešama papildu programmēšanas resursu piesaistīšana, dažkārt tā nav iespējama, vai arī iegūtais rezultāts ir vizuāli nebaudāms informācijas sistēmas un iekļaujamās komponentes krasi atšķirīgā dizaina dēļ. Taču, ja iekļaujamās komponentes lietotāju saskarne ir personalizējama un pielāgojama informācijas sistēmai, kurā to iekļauj, iepriekšminētās problēmas ir iespējams atrisināt.

2.3. Personalizācijas trūkumi

2.3.1.1. Ierobežota anonimitāte

Informācijas sistēmās, kurās ir izstrādātas personalizācijas iespējas, bieži lietotāja pielāgotais izskats tiek saglabāts pie lietotāja profila informācijas (īpaši bieži šāda pieeja ir tīmekļa bāzētām informācijas sistēmām, jo tām ir stingri ierobežota pieeja saglabāt personalizācijas datus lietotāja lokālajos resursos). Tātad, lai izmantotu personalizācijas iespējas, ir nepieciešama lietotāja profila uzturēšana, kas prasa papildu datu uzkrāšanu par lietotāju un lietotāja autentifikāciju. Tāpēc, ne vienmēr personalizācija ir iespējama, ja lietotājs vēlas palikt anonīms.

2.3.1.2. Papildu lietotāja datu nepieciešamība

Lai nodrošinātu personalizācijas iespējas, bieži izstrādātājiem nepieciešams uzkrāt papildu informāciju par lietotāju (dažādus personalizācijas kritērijus skat. 2.3. nodaļu), ko lietotājs ne vienmēr vēlas sniegt. Piemēram, lai izveidotu informācijas sistēmas saskarni, kas atkarīga no

lietotāja dzīves vietā esošajiem laika apstākļiem, ir nepieciešams nolasīt vai pieprasīt no lietotāja viņa dzīves vietas koordinātes.

2.3.1.3. Ātrdarbības zudumi

Lai, lietotāja uzsākot darbu ar sistēmu, programmatūrā ielādētu un apstrādātu uzstādītu lietotāja personalizētos uzstādījumus, ir nepieciešama papildu datu ielasīšana, apstrāde un konfigurācijas uzstādīšana, kas samazina informācijas sistēmas ātrdarbību. Īpaši izteikta šī problēma ir tīmekļa informācijas sistēmām, kurām ir iespējams norādīt lietotāja profila izskatu, mainot vietnes fona attēlus. Šādi fona attēli parasti ir liela izmēra, tāpēc to ielādēšana pārkūlprogrammā var aizņemt pat vairākas sekundes. Turklāt, ņemot vērā, ka katra lietotāja profilā ir dažādi fona attēli, tiek zaudēti pārlūkprogrammas kešatmiņas izmantošanas ieguvumi.

2.4. Personalizācijas kritēriji

Lai informācijas sistēmu pielāgotu individuāli konkrēta lietotāja vai lietotāju grupas vajadzībām, ir nepieciešams iegūt papildu lietotāja informāciju, kuru analizējot tiks veikta personalizācija:

- **Lietotāja piederība noteiktām lietotāju grupām**. Informācijas sistēmas saturs tiek pielāgots, atkarībā no lietotāja piederības konkrētām grupām, attiecīgi šo grupu tiesībām vai interesēm. Bieži vien lietotāju iedalījums grupās tiek veikts drošības dēļ, tāpēc šie procesi ir apskatāmi kopēji.
- **Lietotāja uzdevumi konkrētajā informācijas sistēmā.** Informācijas sistēmas lietotāju saskarne tiek pielāgota konkrēta lietotāja vai lietotāju grupas veicamajiem uzdevumiem sistēmā, ja tie ir atšķirīgi no citu lietotāju uzdevumiem.
- **Lietotāja biežāk veicamās darbības.** Informācijas sistēmas saturs vai elementi, kurus konkrētais sistēmas lietotājs biežāk izmanto savā darbībā, tiek izcelti vai parādīti ar augstāku prioritāti, jo tiek uzskatīts, ka tie satur lietotājam noderīgāku informāciju.
- **Lietotāja gaume.** Informācijas sistēmas lietotāju saskarnes izskats tiek pielāgots lietotāja gaumei – tiek lietotas lietotājam patīkamas krāsas un fonti, izkārtojums u.c.

- **Lietotāja pieredze.** Atkarībā no lietotāja pieredzes darbā ar datoru un darbā ar konkrēto informācijas sistēmu, var mainīties veidi, kā lietotājs paveic konkrēto uzdevumu (piemēram, karsto taustiņu lietošana).
- **Lietotāja vecums.** Atkarībā no lietotāja vecuma, pārsvarā mainās gan lietotājam interesējošā informācija, gan krāsu izjūta, gan spējas ātri pierast pie jauna veida saskarnes elementiem u.c. īpašības.
- **Lietotāja dzīvesvieta.** Atkarībā no lietotāja dzīvesvietas, informācijas sistēmā lietotājam piedāvāta tiek attēlota vai izcelta informācija, kas attiecas uz reģionu, kurā viņš dzīvo.

2.5. Lietotāju saskarnes personalizācijas konfigurācijas uzglabāšana

Lai informācijas sistēmā uzglabātu dažādas iespējamās lietotāju saskarnes personalizētās saskarnes, ir nepieciešams izstrādāt veidu, kā saglabāt lietotāju konfigurāciju. Šajā nodaļā darba autors ir izpētījis un aprakstījis vairākus risinājumus, kas tiek izmantoti citās informācijas sistēmās, personalizācijas konfigurācijas saglabāšanai.

2.5.1. Iepriekš definētu parametru norādīšana

Vienkāršākais veids, kā saglabāt lietotāja personalizētos lietotāju saskarnes uzstādījumus ir norādot iepriekš definētus iestatījumus formātā „parametrs – vērtība”. Šādus parametrus ir iespējams saglabāt datu bāzē vai datņu sistēmā (gan servera datņu sistēmā, gan lokāli,lietotāja datora datņu sistēmā), programmatūras konfigurācijas failā (piemēram, INI failā, XML vai JSON formātā).

Šāds personalizācijas datu saglabāšanas veids tiek izmantots arī tīmekļa pārlūkprogrammas *Google Chrome* tēmu izveidošanai [4]. Lai izveidotu šīs pārlūkprogrammas tēmu, nepieciešams izveidot JSON failu (2.4. att.), kurā tiek norādīti iepriekš definēti konfigurācijas parametri (piemēram, rīkjoslas fona krāsu vai fona attēlu, pogu fona krāsu u.c.), ar kuriem ir iespējams mainīt pārlūkprogrammas loga izskatu.

```
{
    "version" : "2.6",
    "name" : "camo theme",
    "theme" : {
        "images" : {
            "theme_frame" : "images/theme_frame_camo.png",
            "theme_frame_overlay" : "images/theme_frame_stripe.png",
            "theme_toolbar" : "images/theme_toolbar_camo.png",
            "theme_ntp_background" : "images/theme_ntp_background_norepeat.png",
            "theme_ntp_attribution" : "images/attribution.png"
        },
        "colors" : {
            "frame" : [71, 105, 91],
            "toolbar" : [207, 221, 192],
            "ntp_text" : [20, 40, 0],
            "ntp_link" : [36, 70, 0],
            "ntp_section" : [207, 221, 192],
            "button_background" : [255, 255, 255]
        },
        "tints" : {
            "buttons" : [0.33, 0.5, 0.47]
        },
        "properties" : {
            "ntp_background_alignment" : "bottom"
        }
    }
}
```

2.4. att. **Google Chrome tēmas apraksta piemērs**

Lai arī, personalizācijas iestatījumus saglabājot šādā formātā, tos ir vienkārši nolasīt, formāts ir viegli saprotams un rediģējams, taču, konfigurāciju saglabājot šādā formātā, personalizācijas iespēju klāsts ir ierobežots. Programmatūras kodā ir stingri definēta katra konfigurācijas failā esošā parametra apstrāde. Piemēram, šādā veidā nav iespējams papildināt lietotāju saskarni ar kādu jaunu, iepriekš neparedzētu saskarnes elementu. Šo iemeslu dēļ FIBU programmatūras personalizācijas iespēju izstrādei šāds konfigruāciajs datu uzglabāšanas risinājums netika izskatīts.

2.5.2. Šablonu sistēmas

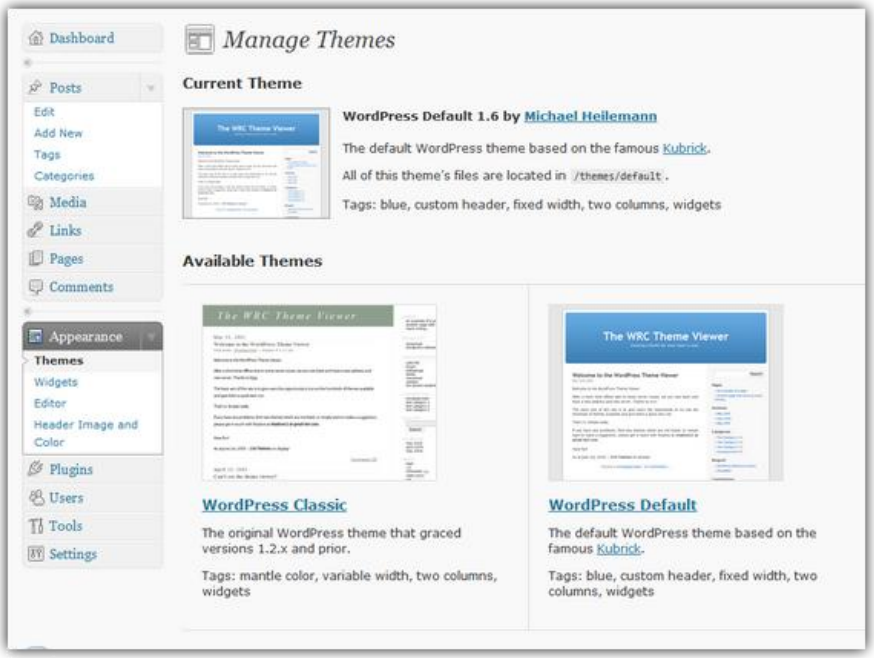
Dažādos tīmekļa vietņu ietvaros un satura pārvaldības sistēmās, lai izveidotu vietnes personalizētu izskatu, ir izveidotas šablonu sistēmas.

Šablons ir no informācijas sistēmas loģikas līmeņa nodalīta koda daļa, kurā tiek definēts tās izskats. Šablonā tiek definēts vietnes statiskie dizaina elementi. Lai šablonā iekļautu vietnes

dinamisko informāciju, tiek norādīti atsevišķi satura bloki, kuros tiek izvietotas lietotāja izvēlētās dinamiska satura komponentes, vai arī šablona koda failos tiek drukāti programmatūras loģikas līmenī sagatavoti vietnes dati. Informācijas sistēmā ir iespējams ievietot vairākus šablonus, no kuriem parasti viens šablons tiek norādīts kā vietnes noklusētais šablons (to norāda vietnes administrators). Vietnes lietotājam var tikt dota iespēja izvēlēties, kuru no šabloniem viņš izmantos, pārskatot vietnes saturu.

Šablonu sistēma ir izveidota arī pasaulē visplašāk izmantotajā tīmekļa satura pārvaldības sistēmā *WordPress*, kura, atsaucoties uz 2010. gada *W3Techs* statistiku [5], ir izmantota 13.7% *W3Techs* pārraudzībā esošo pasaules tīmekļa vietņu veidošanai. Katra *WordPress* tēma sastāv no tajā izmantotajiem attēla failiem, CSS stila failiem un PHP koda failiem, uz kuriem no vietnes loģikas līmeņa, izmantojot MVC principus, tiek nodoti mainīgie ar vietnē attēlojamo saturu [6]. PHP koda failos tiek norādīta vietnes HTML koda struktūra un izdrukāti nepieciešamie mainīgie. *WordPress* tēmas tiek ievietotas no loģikas līmeņa nodalītā apakškatalogā. Vietnes Tēmu pārvaldība notiek *WordPress* administrācijas panelī (2.5. att), caur kuru no vietnē ievietotajām tēmām ir iespējams norādīt vietnes noklusēto tēmu, kā arī iespējams pievienot jaunu tēmu vai dzēst kādu no vietnē iekļautajām tēmām.

WordPress izstrādātāju mājas lapā [7] ir iespējams lejupielādēt vairāk nekā 1000 dažādas gatavas tēmas, taču, vietnes izstrādātāji vienkārši var izveidot arī savām vēlmēm un vajadzībām specifisku tēmu.



2.5. att. **Wordpress tēmu pārvaldība vietnes administrācijas panelī**

Izmantojot šablonu sistēmas, ir iespējams plaši variēt informācijas sistēmas lietotāju saskarnes, taču, lai izveidotu šablonu, ir nepieciešamas pamata tīmekļa vietņu kodēšanas prasmes - HTML, JavaScript un CSS pārzināšana, kā arī nelielas vietnes izstrādes valodas zināšanas. Lai arī ir atrodami trešo pušu izstrādāti grafiski satura pārvaldības sistēmu šablonu veidošanas rīki, piemēram, *Artisteer* [8], taču tie ir maksas produkti, turklāt tie neatbalsta iespēju papildināt rīka funkcionalitēti, dodot iespēju savas izstrādātās informācijas sistēmas tēmu izveidi, kas ļauj saglabāt tēmas iestatījumus konkrētās programmatūras atbilstošajā struktūrā. Turklāt, šablonu sistēmas nav platform-neatkarīgas. Šo faktoru dēļ, šablonu sistēma nav iespējama FIBU programmatūras personalizācijas iespēju izveidei.

2.5.3. Lietotāju saskarnes aprakstīšanas valodas

Lai aprakstītu personalizācijas iestatījumus, ir iespējams izmantot arī dažādas lietotāju saskarnes aprakstīšanas valodas. Lietotāju saskarnes aprakstīšanas valodas ir strukturētas aprakstīšanas valodas, ko izmanto lietotāju saskarnes specificēšanai – lietotāju saskarnē esošo elementu uzskaitīšanai, to stila īpašību definēšanai un saskarnes elementu savstarpējās mijiedarbības norādīšanai [9].

Lietotāju saskarnes aprakstīšanas valoda lietotāju saskarnes personalizācijai ir izmantota multimediju failu atskaņošanas lietotnes *Winamp* 3. versijā. Šīs lietotnes autori *Nullsoft* ir izveidojuši starp-platformu GUI ietvaru *Wasabi,* kas atbalsta personalizācijas risinājumus, izmantojot lietotāju saskarnes aprakstīšanas valodu WasabiXML [10]. Lai izveidotu *Winamp* tēmu, nepieciešams izveidot XML aprakstu, kurā tiek norādīti, kādi satura bloki tiks attēloti šajā tēmā, kā arī tiek norādīti šo satura bloku krāsu toņi un fona attēli. Šis fails kopā ar tēmā izmantotajiem attēliem tiek saarhivēts un, ievietojot to specifiskā programmatūras mapē, programmatūrā ir iespējams to izvēlēties kā izskata tēmu.

Ņemot vērā, ka lietotāju saskarnes aprakstīšanas valodās izveidotie apraksti ir strukturēti teksta faili, izveidot šādu aprakstu renderētāju, kas atbilstoši aprakstam spēj izveidot lietotāju saskarni, ir iespējams praktiski jebkurā programmatūras izstrādes platformā, kas atbalsts grafisku lietotāju saskarņu veidošanu. Tātad, izmantojot lietotāju saskarnes aprakstīšanas valodas, ir iespējams izveidot no platformas neatkarīgu personalizācijas risinājumu.

Pēc darba autora vērtējuma, izpētot vairākas lietotāju saskarnes aprakstīšanas valodas (UIML [11], XAML [12], MXML [13]), tika secināts, ka, izmantojot kādu no esošajām vai izveidojot jaunu lietotāju saskarnes aprakstīšanas valodu, ir iespējams izveidot personalizācijas risinājumu, kas atbilst visām bakalaura darba 1. nodaļā aprakstītajām prasībām, tāpēc tika nolemts izmantot šo risinājumu.

Plašāka informācija par lietotāju saskarnes aprakstīšanas valodu iespējām un to uzbūvi ir iekļauta bakalaura darba 3. nodaļā.

3. LIETOTĀJU SASKARNES APRAKSTĪŠANAS VALODAS

Lietotāju saskarnes aprakstīšanas valodas ir strukturētas aprakstīšanas valodas, ko izmanto lietotāju saskarnes aprakstīšanai. Lietotāju saskarnes aprakstīšanas valodas ir abstraktas valodas, kas nozīmē, ka tajās uzdota tikai vispārīgs saskarnes apraksts, ko, atkarībā no lietotāju saskarnes veidošanai izmantotās platformas, ir iespējams interpretēt dažādi.

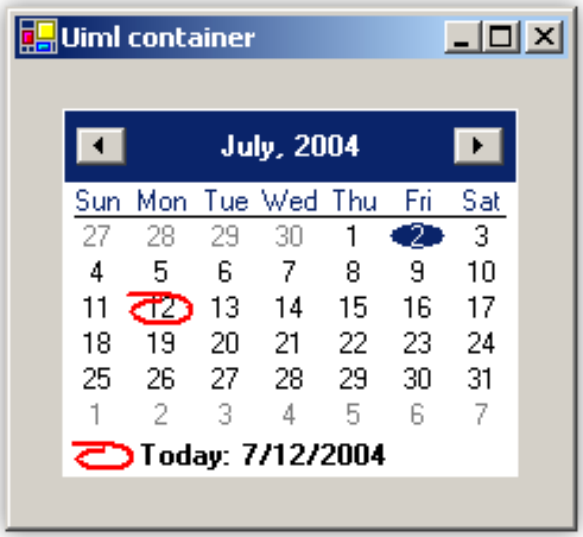
Lietotāju saskarnes aprakstīšanas valodu pirmsākumi ir datējami ar 1997. gadu, kad tika prezentēta pirmā šāda valoda - UIML (termins „lietotāju saskarnes aprakstīšanas valoda" cēlies no šīs valodas nosaukuma).

UIML tika veidota, ievērojot aizvien pieaugošo HTML popularitāti. HTML ļauj lietotājiem bez plašām programmēšanas zināšanām izveidot grafiskas lietotāju saskarnes [11]. Ņemot vērā, ka ne priekš visām programmēšanas valodām ir izveidotas izstrādes vides, kas piedāvā formu dizaina iespējams, tika plānots, ka izmantojot UIML lietotāji varēs vienkāršā veidā izveidot un rediģēt formas, turklāt, formu veidošana būs vienāda visām platformām, kurām ir izveidots UIML interpretators.

UIML ir veidota kā universāla, platformas neatkarīga valoda, ko iespējams pielietot gan darbvirsmas, gan tīmekļa, gan mobilo, gan arī iebūvēto un pat ar balsi vadāmo lietojumprogrammu lietotāju saskarņu aprakstīšanai [14]. Ņemot vērā, ka UIML specifikācija neierobežo aprakstā iekļauto tagu loku, UIML apraksta struktūru ir iespējams papildināt, iekļaujot tajā jaunus elementus un vadīklas, taču attiecīgi šīm izmaiņām ir nepieciešams papildināt visu apraksta renderētāju funkcionalitāti, iekļaujot tajos šo elementu atbalstu.

UIML renderētāji ir izveidoti vairāk nekā 10 dažādām platformām, piemēram, C++, .NET, Java, HTML, Visual Basic [15].

1. pielikumā ir dots UIML apraksta piemērs. Pielikumā dotajam aprakstam atbilstoša lietotāju saskarne, kas renderēta izmantojot UIML.net (UIML renderētāju, kas izveidots C# valodā, izmantojot .NET ietvaru [16]) parādīta 3.1. att.



3.1. att. **Forma, kas renderēta pēc UIML apraksta**

UIML aprakstā atsevišķi tiek definēti lietotāju saskarnes elementi (tie hierarhiski uzskaitīti aprakstā zem taga *structure*) un to īpašības. Katram elementam kā atribūti tiek norādīts tā identifikators (*id*) un klase (*class*) jeb elementa tips (piemērā uzdoti divi elementi – ar tipu *Container*, kas tiek renderēts kā logs, un ar tipu *Calendar*, kas tiek renderēts kā kalendāra vadīkla). Saskarnes elementu konkrētas izskata īpašības ir uzdotas zem taga *style*. Katra īpašība uzdota kā atsevišķs tags *property*, kā atribūtus norādot atbilstošā elementa identifikatoru (*part-name*), bet kā taga vērtība ir norādīta konkrētās īpašības vērtība. Piemērā dotajā aprakstā šādā veidā tiek norādīti loga izmēri, kā arī kalendāra izmēri, pozīcija, iezīme, kā arī mazākais iespējamais datums un lielākais iespējamais datums.

3.1. Lietotāju saskarnes apraksta struktūra

Izstrādājot informācijas sistēmas lietotāju saskarni, ir iespējams izšķirt dažādus lietotāju saskarnes abstrakcijas līmeņus.

- **Uzdevumu līmenis.** Augšējais abstrakcijas līmenis – uzdevumu līmenis – apraksta, kādus uzdevumus lietotājam nepieciešams paveikt informācijas sistēmā, izmantojot konkrēto lietotāja saskarni.

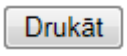
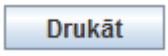
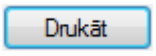
- **Abstrakts lietotāju saskarnes līmenis.** Šis abstrakcijas līmenis iekļauj informāciju par to, kādus lietotāja saskarnes elementus (saskarnes blokus, vadīklas) un kādus datus nepieciešams iekļaut saskarnē, lai lietotājs spētu paveikt viņam nepieciešamos uzdevumus.
- **Konkrēts lietotāju saskarnes līmenis.** Šajā abstrakcijas līmenī ir uzskaitītas lietotāju saskarnē iekļauto elementu konkrētas izskata īpašības, kas atšķiras katram saskarnes elementa tipam. Piemēram, saskarnes elementam – vadīklai komandpoga parasti ir iespējams norādīt īpašības *iezīme, augstums, platums, fona krāsa, ikona, robežas krāsa, robežas platums* u.c..
- **Gala lietotāja saskarnes līmenis.** Reprezentē lietotāju saskarnes elementu ar tā specifiskajām izskata īpašībām konkrētā programmatūras platformā, konkrētā programmēšanas valodā.

Parasti lietotāju saskarnes aprakstos ir iespējams specificēt lietotāja saskarni līdz trešajam abstrakcijas līmenim, taču, tas ir atkarīgs gan no izmantotās apraksta valodas, gan arī no tā, kādiem mērķiem un uz kādām platformām paredzēts darbināt konkrēto informācijas sistēmu, jo ne visas platformas atļauj plašas iespējas saskarnes elementu noformēšanai.

Apskatīsim iepriekš rakstīto ar konkrētu piemēru. Informācijas sistēmā ir nepieciešama spiedpoga, kas izsauc rēķina drukāšanu. Atkarībā no platformas un informācijas sistēmas izstrādes vides, ir iespējamas dažādas interpretācijas šādam elementam (3.1. tabula). Tabulā ir attēlota spiedpogas realizācija tīmekļa vidē (HTML), Java programmēšanas valodā, izmantojot *Swing* GUI bibliotēku, C# vidē, izmantojot .NET ietvaru, kā arī implementācija fiziskai iekārtai.

3.1. tabula

Lietotāju saskarnes apraksta interpretācija dažādām platformām

	HTML	Java Swing	C# .NET	Fiziska iekārta
Uzdevumu līmenis	Rēķinu drukāšana			
Abstrakts saskarnes līmenis	Spiedpogas vadīkla			Fiziska spiedpoga
Konkrēts saskarnes līmenis	Grafiska 2-dimensionāla spiedpoga. Iezīme - „Drukāt”.			Fiziska plastikāta spiedpoga.
Gala lietotāju saskarnes līmenis	<input type="button" value="Drukāt" /> 	JButton btn = **new** JButton("Drukāt"); 	Button btn = **new** Button(); btn.Text="Drukāt"; 	

Redzam, ka atkarībā gan lietotnes izstrādes platformas piemērā minētās lietotāju saskarnes elementa interpretācija var būtiski atšķirties. Izmantojot lietotāju saskarnes aprakstīšanas valodas, ir iespējams izveidot lietotāju saskarnes aprakstu, kas specificē piemērā minētās saskarnes elementus. Šādu aprakstu iespējams interpretēt visās piemērā minētajās platformās, iegūstot dažādas lietotāju saskarnes, ko iespējams izmantot viena uzdevuma paveikšanai.

3.2. Lietotāju saskarnes renderēšana

Lietotāju saskarnes renderēšana ir process, kas konvertē lietotāju saskarnes aprakstu gala lietotājam redzamā formā, izmantojot kuru, lietotājs mijiedarbosies ar informācijas sistēmu [14]. Renderētājā ir jānorāda, kāds lietotāju saskarnes elements tiks izveidots katra apraksta elementa vietā un kā notiks elementa atribūtu uzstādīšana. Renderēšanu var būt realizēta divos veidos:

- **Kompilējoša renderēšana.** No lietotāju saskarnes apraksta tiek ģenerēts mašīnkods vai kods kādā programmēšanas valodā, kas tiek kompilēts kopā ar pārējo programmatūras kodu. Šajā kodā ir iekļauta konkrētajam aprakstam atbilstošas lietotāju saskarnes veidošana, kā arī saskarnes elementu mijiedarbības apstrādes funkcijas.
- **Interpretējoša renderēšana.** Programmatūrā ir iekļauts kods, kas programmatūras darbības (visticamāk, ielādes) laikā ielasa atbilstošo lietotāju saskarnes aprakstu, parsē to un dinamiski izveido aprakstam atbilstošu lietotāju saskarni, kā arī dinamiska saskarnes elementu mijiedarbības norādīšana.

Lai izveidotu personalizācijas risinājumu, kas lietotāju saskarnes konfigurāciju saglabā, izmantojot kādu lietotāju saskarnes aprakstīšanas valodu, nepieciešams izmantot interpretējošo renderēšanu, jo tā neprasa papildu kompilāciju pēc personalizācijas veikšanas.

3.3. Lietotāju saskarnes aprakstīšanas valodu salīdzinājums

No autora apskatītajām pašreiz pielietotajām lietotāju saskarnes aprakstīšanas valodām (3.2. tabula), visas ir XML saimes valodas. Lai arī UIML, kas tika izlaista 1997. gadā, tika prezentēta kā universāla un derīga jebkuras saskarnes aprakstīšanai, kopš tā laika ir izlaistas vairākas domēn-specifiskas valodas, kas paredzētas specifiskām platformām vai konkrētai programmatūrai, aprakstu struktūru pielāgojot un optimizējot, izmantojot šai platformai specifiskus lietotāju saskarnes elementus un to īpašības. Tabulā ir uzskaitīts apskatīto valodu pielietojums, to renderēšanas veids, platforma, kurā ir izveidots šīs valodas interpretators, kā arī valodas saime.

3.2. tabula

Lietotāju saskarnes aprakstīšanas valodu salīdzinājums

Nosaukums	Pielietojums	Renderēšanas veids	Interpretējošā platforma	Valodu saime
UIML [11]	Dažādi	Dažādi	Dažādi	XML
UsiXML [17]	Dažādi	Dažādi	Dažādi	XML
MXML [13]	Adobe Flex	Kompilējoša	Actionscript	XML
XAML [12]	WPF, Silverlight	Kompilējoša	.NET valodas, JavaScript	XML
OpenLaszlo [18]	Adobe Flash	Kompilējoša	ECMAScript	XML
XUL[19]	Mozilla Firefox paplašinājumi	Interpretējoša	JavaScript	XML
WTKX [20]	Apache Pivot programmas	Interpretējoša	Java	XML
WasabiXML [10]	Winamp tēmu veidošana	Interpretējoša	C / C++	XML

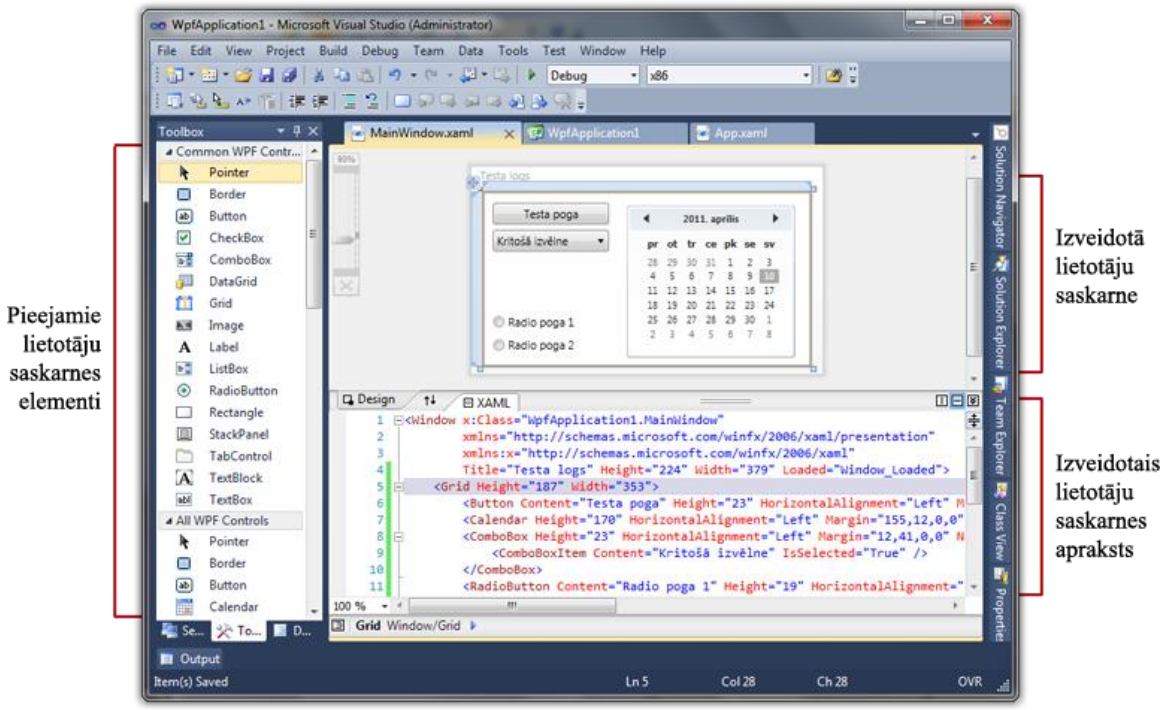
3.4. Lietotāju saskarnes apraksta veidošanas rīki

Lai atvieglotu lietotāju saskarnes aprakstu izveidošanu, kā arī ļautu aprakstu izveidot cilvēkiem, kas nepārzina izmantoto lietotāju saskarnes aprakstīšanas valodu, tiek izstrādāti grafiski lietotāju saskarnes redaktori. Redaktoros ir iespējams vienkārši rediģēt lietotāja saskarnes elementus un to izskatu, pārsvarā, izmantojot *drag-and-drop* un WYSIWYG pieejas.

Lietotāju saskarnes apraksta redaktori var būt gan iebūvēti programmatūrā, kuras saskarnes apraksts tiek veidots, gan iebūvēti programmatūras izstrādes vidē, gan arī neatkarīgi rīki. Lietotāju saskarnes apraksta redaktori var būt gan pielāgoti kādai konkrētai saskarnes implementācijai (piemēram, saskarne *Java* valodā, izmantojot *Swing* GUI rīkkopu), gan arī

neatkarīgi no konkrētas implementācijas, kas, iespējams, neatbalsta pilnu apraksta funkcionalitāti.

Lietotāju saskarnes apraksta veidošanai aprakstīšanas valodā XAML ir *Microsoft Visual Studio* izstrādes vidē integrēts grafiskais redaktors (3.2. att). Izmantojot šo izstrādes vidi, lietotāju saskarnes aprakstu ir iespējams mainīt gan grafiski, izmantojot formu redaktoru, gan rediģējot apraksta tekstu, iekļaujot aprakstā jaunus elementus un rediģējot šo elementu izskata īpašības.



3.2. att. **XAML lietotāju saskarnes apraksta redaktors *Microsoft Visual Studio* izstrādes vidē**

4. IZSTRĀDĀTAIS RISINĀJUMS

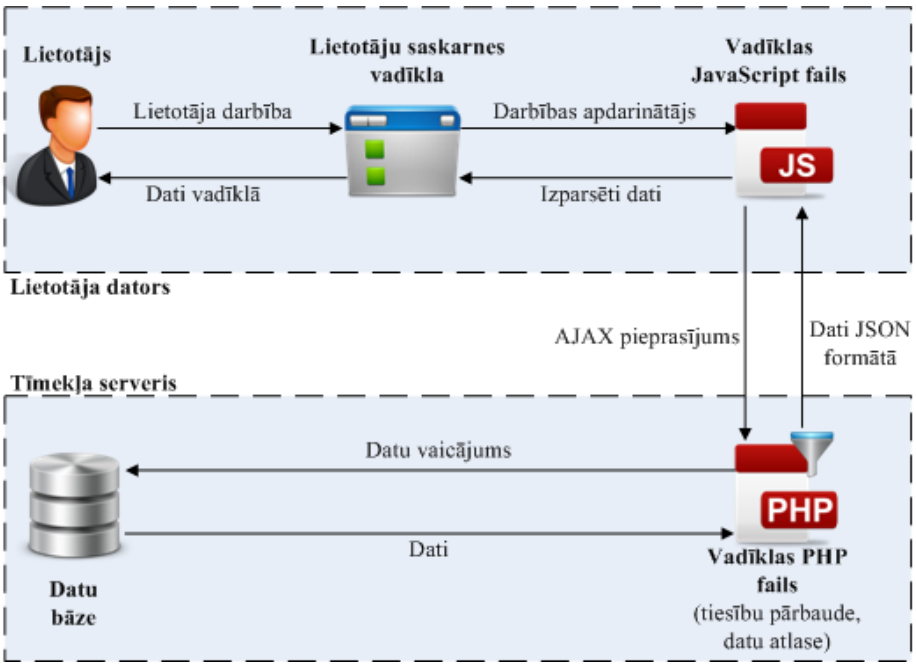
Kā minēts bakalaura darba 1. nodaļā, sākotnēji personalizācijas risinājums bija jāizstrādā tikai FIBU programmatūras tīmekļa versijā. Darba rakstīšanas brīdī ir izstrādāts personalizācijas risinājums, kas lietotāju saskarnes konfigurācijas saglabāšanai izmanto FIBU izstrādātāju izveidotu lietotāju saskarnes aprakstīšanas valodu.

4.1. FIBU tīmekļa versijas lietotāju saskarnes bibliotēka

2009. gadā, uzsākot veidot FIBU programmatūras tīmekļa versiju, tā tika veidota pēc iespējas līdzīga programmatūras darbvirsmas versijai, kas izveidota Microsoft Visual Basic 6 vidē.

Ņemot vērā, ka tīmekļa vidē ir ļoti minimāls iebūvēto vadīklu klāsts (teksta ievades lauks, izvēlnes rūtiņa, radio poga, izkrītošās izvēlne un spiedpoga), izstrādājot FIBU programmatūras tīmekļa versiju, lai tajā nodrošinātu darbvirsmas versijā esošās iespējas, tika izveidota JavaScript lietotāju saskarnes bibliotēka. Šajā bibliotēkā ir izveidotas līdzīgas vadīklas, kādas ir izmantotas programmatūras darbvirsmas versijā, piemēram, vadīkla kokveida struktūras attēlošanai un pārvaldībai, vadīkla datuma izvēlnei u.c.. Tā kā FIBU programmatūrā (gan datu ievades modulī, gan programmatūras administrēšanas modulī) bieži ir nepieciešami konkrētajam risinājumam specifiski šo vadīklu paveidi (piemēram, kokveida struktūrā attēlotas lietotājam pieejamās iestādes), JavaScript bibliotēkā ir iekļautas arī šādas, specifiskas vadīklas, kuras manto vispārīgo vadīklu īpašības, bet tajās jau ir iestrādātas konkrētiem mērķiem pielāgotas specifiskas īpašības un metodes, kā arī jau ir nodrošināta šim lietojumam specifiskas datu atlases.

Lai atlasītu šajās vadīklās attēlojamos programmatūras datus (piemēram, organizāciju struktūru), tiek izmantotas AJAX tehnoloģijas. Reaģējot uz lietotāja darbībām, kas prasa papildu datu atlasi un attēlošanu, uz serveri tiek nosūtīts AJAX pieprasījums. Servera pusē, apstrādājot šo pieprasījumu, notiek lietotāja tiesību pārbaude un lietotāja tiesībām atbilstošu pieejamo datu atlase. Kā AJAX pieprasījuma atbilde uz vadīklu tiek nosūtīti atlasītie dati serializēti JSON formātā, kur tie tiek apstrādāti un attēloti lietotājam saprotamā formā (4.1. att.).



4.1. att. **Datu atlase FIBU bibliotēkas vadīklām**

Tāpat kā HTML noklusētās vadīklas, arī FIBU bibliotēkas vadīklas izraisa dažādus notikumus (piemēram, kokveida struktūras vadīklai notikums *onSelect*, kas tiek izraisīts, kad lietotājs izvēlas kādu no struktūras elementiem, uzspiežot tam ar peles kreiso taustiņu). Šos notikumus ir iespējams apstrādāt, izmantojot JavaScript programmēšanas valodu, katram notikumam piesaistot vienu vai vairākus notikumu apdarinātājus. Notikumu apdarinātāja funkcijai tiek nodoti konkrētajam notikumam specifiski parametri (piemēram, kokveida struktūras vadīklas notikums *onSelect* kā parametrus nodod izvēlētā elementa objektu un norādi uz konkrēto vadīklu).

4.2. Izveidotā lietotāju saskarnes aprakstīšanas valoda

Tā kā programmatūras tīmekļa versijā lietotāju saskarnē izmantoto vadīklu veidošana notiek, izmantojot JavaScript programmēšanas valodu, arī lietotāju saskarnes apraksta renderētājs tika veidots JavaScript valodā. Tāpēc lietotāju saskarnes apraksta valodai ir jābūt tādai, ko pēc iespējas vieglāk un ātrāk var parsēt ar JavaScript, taču jābūt iespējams to parsēt arī, izmantojot citas programmēšanas valodas (primāri – Visual Basic 6, kurā ir izveidota FIBU programmatūras darbvirsmas versija).

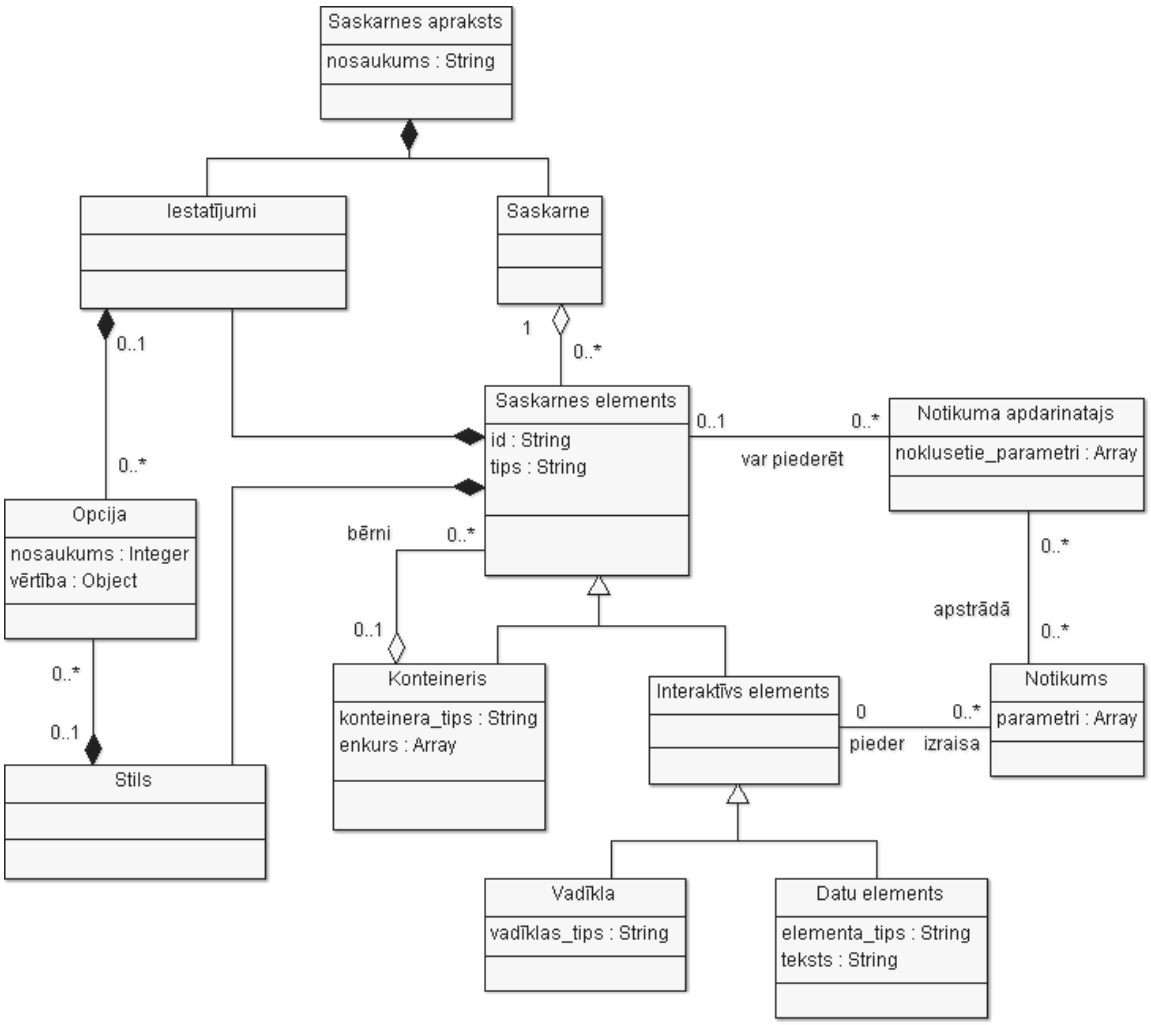
Kā potenciālie apraksta formāti tika apskatīti XML un JSON, kas ir tīmekļa vidē visplašāk izmantotie datu serializēšanas formāti. Lai salīdzinātu šo formātu parsēšanas ātrdarbību un ielasīšanas ātrumu JavaScript valodā, tika izveidots testa skripts, kas parsē kā objektu bakalaura darba 2. pielikumā parādītos JSON un XML formāta aprakstus, kuros ir uzskaitīti vienādi personu aprakstoši parametri. JSON apraksta parsēšanai tika izmantota *Prototype* bibliotēkas funkcija *String.evalJSON()*, jo FIBU informācijas sistēmā tiek pielietota šī bibliotēka. XML apraksta parsēšai tika izveidota JavaScript iebūvētā klase *DOMParser*. Salīdzinot šo aprakstu parsēšanas ātrdarbību uz dažādām pārlūkprogrammām, JSON apraksta parsēšanas skripts darbojās 10-20 reizes ātrāk, turklāt, izmantojot iepriekšminēto bibliotēkas funkciju, JSON apraksta parsēšanu izveidot ir daudzkārt vienkāršāk.

Tā kā izstrādāto lietotāju saskarnes aprakstīšanas valodu bija plānots izmantot tikai FIBU programmatūras personalizācijas risinājumam, to bija iespējams izveidot vienkāršāku un konkrētajam risinājumam piemērotāku nekā citas darba autora apskatītās lietotāju saskarnes aprakstīšanas valodas.

Šo iemeslu dēļ tika nolemts veidot jaunu lietotāju saskarnes aprakstīšanas valodu, kura izmanto JSON notāciju.

4.2.1. Saskarnes apraksta metamodelis

FIBU personalizācijas risinājumā izmantotās lietotāju saskarnes aprakstīšanas valodas metamodelis parādīts 4.2. att.



4.2. att. **FIBU programmatūrā izmantotā saskarnes apraksta metamodelis**

FIBU programmatūras lietotāju saskarnes apraksts sastāv no šādām sastāvdaļām:

- **Saskarnes apraksts.** Lietotāju saskarnes apraksts (saknes elements);
- **Iestatījumi.** Dažādi konkrētās lietotāju saskarnes iestatījumi, piemēram, pārlūkprogrammas loga nosaukums;
- **Saskarne.** Visu lietotāju saskarnes elementu kopa, kas veido lietotāju saskarni;
- **Opcija.** Jebkura saskarnes vai konkrēta saskarnes elementa opcija formātā „parametrs – vērtība";
- **Stils.** Saskarnes elementa konkrētu izskata īpašību kopa;
- **Saskarnes elements.** Jebkurš lietotāju saskarnes elements;

- **Konteineris.** Lietotāju saskarnes bloks, kurā paredzēts ievietot citus lietotāju saskarnes elementus, loģiski tos nodalot no pārējiem elementiem. Konteineri ir gan viendaļīgi, gan vairākdaļīgi, ar iespējām lietotājam mainīt šo daļu izmērus;
- **Interaktīvs elements.** Lietotāju saskarnes interaktīvie elementi, kas izraisa dažādus notikumus, reaģējot uz lietotāja veiktajām darbībām;
- **Datu elements.** Lietotāju saskarnes elements vienkāršas, nestrukturalizētas informācijas attēlošanai, piemēram, teksta lauks, attēls, saite;
- **Vadīkla.** Lietotāju saskarnes interaktīvs elements, kas ļauj lietotājam sadarboties ar programmatūru un kas ir paredzēta konkrēta uzdevuma paveikšanai;
- **Notikums.** Notikums, ko izsauc lietotāja darbība ar kādu saskarnes elementu. Notikumus var izsaukt gan konkrēts lietotāju saskarnes elements, gan kāda cita iepriekš definēta darbība programmatūrā (piemēram, notikums „START", kas tiek izsaukts, uzsākot darbu ar programmatūru);
- **Notikumu apdarinātājs.** Programmatūras koda fragments, kas tiek izsaukts, reaģējot uz kādu notikumu. Notikumu apdarinātājs var būt gan konkrēta saskarnes elementa darbība, gan arī kāda programmatūras darbība, kas nav piesaistīta konkrētam elementam. Uz notikumu apdarinātāju var tikt nodoti noklusētie parametri.

4.3. Lietotāju saskarnes apraksta ielasīšana un renderēšana

FIBU programmatūras lietotāju saskarnes apraksti glabājas tīmekļa servera datņu sistēmā. Tie ir novietoti neatkarīgi no programmatūras loģikas līmeņa, tādējādi dažādām programmatūras instancēm ir iespējams pievienot dažādu aprakstu kopu. Katrs apraksts tiek ievietots atsevišķā apakšmapē.

Ielādējot FIBU programmatūru, izmantojot PHP, attiecīgais lietotāju saskarnes apraksts tiek ielasīts no datņu sistēmas un tiek nodots apraksta renderēšanai, kas izveidota JavaScript programmēšanas valodā. Renderēšanas brīdī apraksts tiek parsēts un lietots kā JavaScript objekts.

Sākotnēji tiek uzstādīti lietotāju saskarnes iestatījumi (piemēram, ja šāda opcija ir norādīta, tiek uzstādīts programmatūras loga nosaukums). Katrai iestatījumu opcijai renderētāja kodā ir izveidota šai opcijai specifiska apstrāde.

Pēc lietotāju saskarnes iestatījumu ielasīšanas tiek uzsākta saskarnes elementu veidošana. Aprakstā iekļautās vadīklas un konteineri tiek renderētas kā FIBU JavaScript bibliotēkas objekti, bet datu elementi tiek interpretēti kā HTML elementi. Tā kā lietotāju saskarnes apraksts ir hierarhisks, elementu renderēšana ir rekursīva - ja elements ir konteineris un tam ir norādīti apakšelementi, tie rekursīvi tiek izveidoti un pievienoti konteinerim.

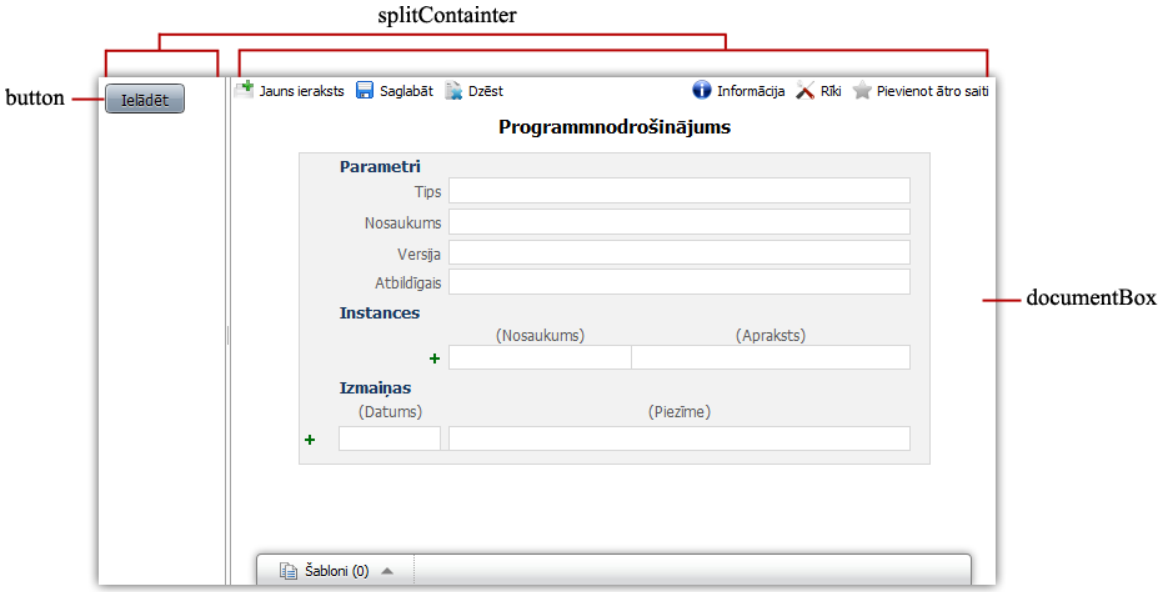
Kad ir izveidoti lietotāju saskarnes elementu, tiek uzstādīti notikumu apdarinātāji. Katrs notikums aprakstā tiek apzīmēts formātā *[elementaID]:notikumaNosaukums*, kur *elementaID* ir notikumu izsaucošā elementa unikāls identifikators (dažādiem speciāliem notikumiem, piemēram, notikumam *START*, kas tiek izsaukts pēc apraksta renderēšanas, identifikators netiek norādīts). Notikumu apdarinātāji aprakstā tiek apzīmēti formātā *[elementaID]:funkcijasNosaukums*. Pēc elementu unikālajiem identifikatora tiek iegūta norāde uz elementam atbilstošo objektu. Norāde uz notikumu apdarinātāju funkciju tiek iegūta, izmantojot *reflection* iespējas. Pierakstīšanās uz elementa notikumiem tiek veikta, izmantojot elementam atbilstošā objekta funkciju *Element.observe(notikums, apdarinatajaFunkcija)* (visas FIBU lietotāju bibliotēkas vadīklas manto šīs funkcijas implementāciju no vienas bāzes klases, bet datu elementiem, kas saskarnē tiek interpretēti kā vienkārši HTML elementi šāda funkcija ir pieejama, izmantojot *Prototype* JavaScript bibliotēku [21]), uz šo funkciju nododot notikuma nosaukumu un palīgfunkciju, kas pēc notikuma izraisīšanas izsauc notikumu apdarinātāja funkciju ar apdarinātāja noklusētajiem parametriem (ja tādi ir nodoti).

Tā kā ne vienmēr notikumu apdarinātāju funkciju ienākošie parametri ir savietojami ar parametriem, kas tiek nodoti, izraisot notikumu, vai arī, ja kā notikumu apdarinātājs ir jāizsauc kāda neatkarīga darbība, dažkārt ir nepieciešams izveidot palīgfunkcijas, kas veic parametru translēšanu vai specifisku darbību izsaukšanu. Tāpēc līdz ar lietotāju saskarnes aprakstu ir iespējams pievienot arī JavaScript koda failu, kurā implementētas nepieciešamās palīgfunkcijas. Šis fails tiek novietots programmatūras tīmekļa servera datņu sistēmā vienā apakšmapē ar

lietotāju saskarnes apraksta failu un programmatūras ielādes laikā tiek ielasīts tīmekļa pārlūkprogrammā.

4.4. FIBU lietotāju saskarnes piemērs

Lai nodemonstrētu konkrētu saskarnes apraksta piemēru, FIBU programmatūrā ir izveidota vienkārša lietotāju saskarne (4.3. att.).



4.3. att. **Demonstrācijas nolūkiem izveidotā testa saskarne**

Šī saskarne sastāv no sadalošā konteinera (*splitContainer*), kas to vertikāli sadala divās daļās un divām vadīklām – spiedpogas (*button*) un veidlapas (*documentBox*)(veidlapa lietotāju saskarnes aprakstā tiek iekļauta kā vienota vadīkla, izmantojot kuru, var dinamiski ielādēt jebkuru veidlapu. Veidlapu izskata rediģēšana tiek veikta, izmantojot FIBU programmatūras veidlapu definēšanas rīku, kas ir neatkarīgs no aprakstītā lietotāju saskarnes redaktora).

3. pielikumā ir dots šai lietotāju saskarnei atbilstošais lietotāju saskarnes apraksts.

Lietotāju saskarnes elementi hierarhiski ir uzskaitīti apraksta sadaļā *layout*. Kā saskarne augšējais līmenis aprakstā ir sadalošais konteineris ar ID *mainContainer* un tipu *splitContainer*. Lai konteineris aizņemtu visu saskarnes logu, tam ir norādīti enkuri pie visām loga malām un

attālumi 0 pikseļi no tām. Sadalošajam konteinerim sadaļā *containers* ir uzskaitīti tā apakškonteineri, respektīvi, katra no daļām, ko sadala augšējais konteineris tiek uzdota kā atsevišķs konteineris.

Pirmajam konteinerim aprakstā tiek norādīts viens tā apakšelements (sadaļā *childs*) – elements ar tipu *control* un vadīklas tipu *button*, kas saskarnē tiek interpretēts kā spiedpogas vadīkla, kas saskarnē redzama kreisajā augšējā stūrī. Spiedpogai aprakstā tiek norādīta tās iezīme (*label*) – „Ielādēt".

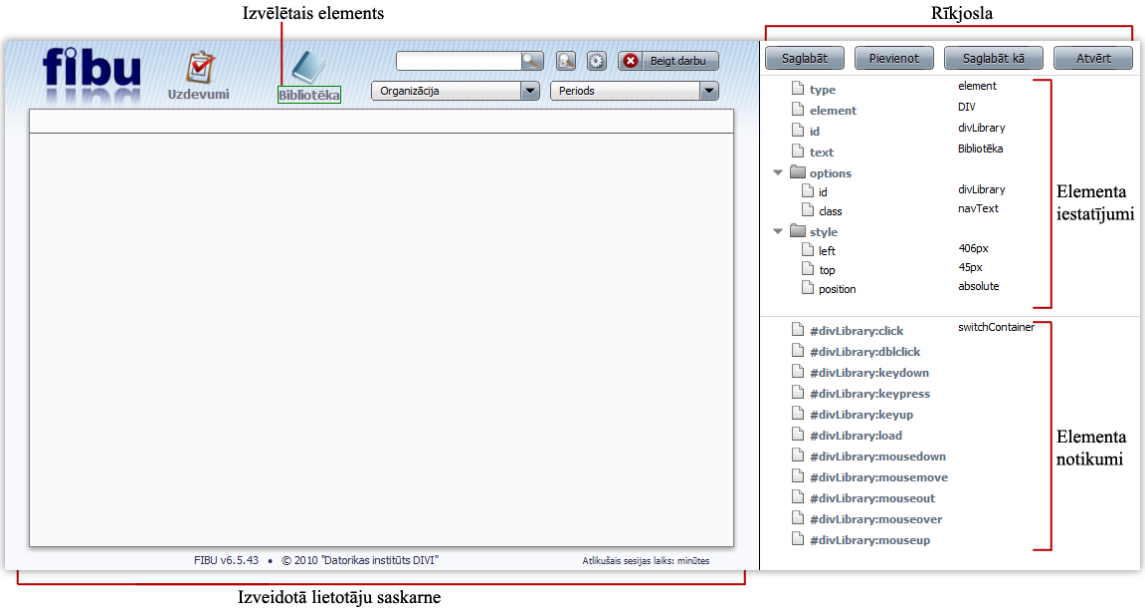
Arī otrajam konteinerim aprakstā tiek norādīts viens bērns – elements ar tipu *control* un vadīklas tipu *documentBox*, kas tiek interpretēts kā saskarnes bloks, kurā tiek attēlota veidlapa (redzama saskarnes labajā pusē).

Saskarnes elementu savstarpējā mijiedarbība tiek norādīta apraksta sadaļā *triggers*. Piemērā minētajā aprakstā tiek uzdota viena šāda apstrāde – reaģējot uz spiedpogas *btnLoad* notikumu *click*, tiks izsaukta veidlapas vadīklas funkcija *load*, tai kā parametrus nododot objektu ar ielādējamās veidlapas aprakstošajām dimensijām (piemērā dotajā aprakstā kā vienīgā dimensija tiek nodots veidlapas identifikators 40177). Šāda mijiedarbība nozīmē, ka, lietotājam nospiežot peles kreiso taustiņu uz saskarnē esošo spiedpogu, tiks ielādēta veidlapa ar identifikatoru 40177 – „Programmnodrošinājums".

4.5. Lietotāja saskarnes redaktors

Lai atviegloti rediģētu personalizēto lietotāju saskarņu aprakstus, FIBU programmatūrā ir izstrādāts lietotāju saskarnes redaktors. Redaktors ir izveidots tīmekļa vidē, izmantojot JavaScript programmēšanas valodu pamata funkcionalitātes nodrošināšanai un PHP programmēšanas valodu nepieciešamajām servera puses darbībām – apraksta ielasīšanai no datņu sistēmas un apraksta saglabāšanai datņu sistēmā.

Lietotāju saskarnes redaktors sastāv no četriem galvenajiem blokiem – lietotāju saskarnes bloka, rīkjoslas, elementa iestatījumu bloka un elementa notikumu bloka (4.4. att).



4.4. att. **FIBU programmatūras lietotāju saskarnes redaktors**

Lietotāju saskarnes blokā tiek attēlots, kā pašreiz izveidotā saskarne tiks attēlota programmatūras tīmekļa versijā (darbojoties ar redaktoru, vadīklās dati netiek ielasīti). Lietotāju saskarnes blokā ir iespējams mainīt saskarnes elementu pozīciju, izmantojot *drag and drop* principu. Ar kreiso peles taustiņu uzspiežot kādam saskarnes elementam, elementa iestatījumu blokā tiek ielasīti izvēlētā elementa iestatījumi, bet elementa notikumu blokā – visi izvēlētā elementa izraisītie notikumi un tiem uzstādītie notikumu apdarinātāji. Elementa iestatījumu blokā ir iespējams mainīt iestatījumu vērtības, kā arī pievienot jaunus elementa iestatījumus. Elementa notikumu blokā ir iespējams pievienot elementa notikumiem atbilstošus notikumu apdarinātājus.

Lietotāju saskarnes redaktora rīkjoslā ir iespējams izsaukt četras darbības. Nospiežot spiedpogu *Saglabāt*, tiek izsaukta atbilstošā lietotāju saskarnes apraksta saglabāšana. Nospiežot spiedpogu *Pievienot*, tiek atvērts uznirstošais logs, kurā iespējams izvēlēties un pievienot pašreiz izvēlētajam konteinerim jaunu apakšelementu. Nospiežot spiedpogu *Saglabāt kā*, lietotāju saskarnes aprakstu ir iespējams saglabāt ar jaunu nosaukumu, dodot iespēju vienā sistēmas instancē lietotājiem lietot vairākas dažādas lietotāju saskarnes. Nospiežot spiedpogu *Atvērt*, tiks piedāvāta iespēja atvērt kādu citu sistēmā definētu lietotāju saskarnes aprakstu rediģēšanai.

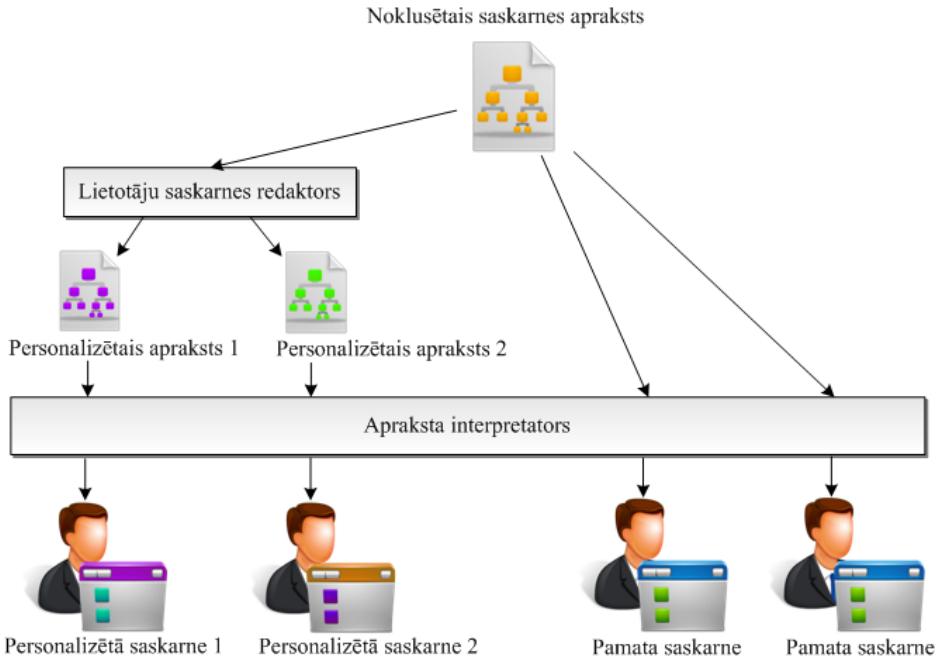
Lietotāju saskarnes redaktora ielādes laikā lietotāju saskarnes apraksts tiek ielasīts un renderēts, izmantojot 4.3. nodaļā aprakstītos principus. Papildus tam, pēc elementu unikālajiem

identifikatoriem tiek noglabātas arī norādes uz vietu aprakstā (t.i. objektā, ko izveido atbilstoši aprakstam), kurā tiek definēti konkrētā elementa iestatījumi. Lietotāju saskarnes redaktorā izmainot kādu no elementa opcijām pēc elementa identifikatora tiek piekļūts šai vietai elementa aprakstā un arī aprakstā tiek izmainīta šī īpašība. Šajā brīdī elements tiek pārrenderēts un lietotāju saskarnes blokā tiek parādīts pārrenderētais elements, kas izveidots, ņemot vērā aprakstā veiktās izmaiņas.

Saglabājot aprakstu, aprakstam atbilstošais JavaScript objekts, izmantojot *Prototype* bibliotēkas iespējas, tiek serializēts kā JSON simbolu virkne. Šī sombolu virkne, izmantojot AJAX tehnoloģijas, tiek nosūtīta uz tīmekļa serveri, kur izmainītas apraksts tiek noglabāts datņu sistēmā.

4.5.1. Lietotāja saskarnes redaktors programmatūras gala lietotājiem

Lai arī FIBU informācijas sistēmas gala lietotāji varētu izveidot personalizētu programmatūras saskarni, nākotnē ir plānots izveidot gala lietotājiem paredzētu lietotāju saskarnes apraksta redaktora versiju ar ierobežotām lietošanas tiesībām. Gala lietotājiem nebūs iespēja izveidot jaunu lietotāju saskarni, bet tikai rediģēt viņiem pieejamo lietotāju saskarnes aprakstu, mainot saskarnes elementu konkrētas stila īpašības (piemēram, elementu izmērus, elementu fona krāsu u.t.t.), un varēs norādīt, kuru no vadīklām viņi izmantos, ja kāda no lietotāju uzdevumu veikšanai ir pieejamas savstarpēji aizstājamas vadīklas. FIBU programmatūras gala lietotāji nevarēs mainīt elementu mijiedarbību, kā arī nevarēs pievienot jaunus vai dzēst esošos elementus. Ja FIBU lietotājs būs izveidojis personalizētu lietotāju saskarni, viņam tiks attēlota personalizētā saskarne, bet, ja viņš saskarni nebūs personalizējis – tiks piedāvāta noklusētā saskarne (4.5. att.).



4.5. att. **FIBU personalizācijas koncepcija gala lietotājiem**

5. REZULTĀTI

Izstrādājot bakalaura darbu darba autors ir izpētījis dažādus informormācijas sistēmu personalizācijas veidus, personalizācijas ieguvumus un trūkumus. Lai izveidotu personalizācijas risinājumu FIBU informācijas sistēmā, darba autors ir apskatījis dažādus veidus, kā izveidot, aprakstīt un uzglabāt personalizācijas konfigurāciju. Par vispiemērotāko veidu konkrētā risinājuma izveidei tika atzīts lietotāju saskarnes aprakstīšanas valodu lietošanu, tāpēc autors ir plašāk izpētījis dažādu aprakstīšanas valodu iespējas un pielietojumos, to struktūru un uzbūvi.

Izmantojot iegūtās zināšanas, darba autors FIBU informācijas sistēmā ir izveidojis personalizācijas risinājumu, kas izmanto jaunu, autora izveidotu lietotāju saskarnes aprakstīšanas valodu. Lai atvieglotu lietotāju saskarnes personalizāciju, ir izveidots lietotāju saskarnes redaktors, ar kura palīdzību ir iespējams ērti un intuitīvi rediģēt saskarnei atbilstošo aprakstu. Izstrādātais personalizācijas risinājums ir ieviests FIBU programmatūras 6.5 versijā un veiksmīgi adaptēts ekspluatācijas vidē - izmantojot to, dažādu klientu vajadzībām tiek pielietotas vismaz 5 atšķirīgas konkrētajam produktam pielāgotas lietotāju saskarnes.

Nākotnē ir plānots attīstīt FIBU programmatūras personalizācijas risinājumu, veicot šādus uzlabojumus:

- Personalizācijas risinājumu izstrādāt arī programmatūras darbvirsmas versijai;
- Izveidot 4.5.1. nodaļā aprakstīto konceptu personalizācijas risinājuma izmantošanai programmatūras gala lietotājiem;
- Papildināt JavaScript lietotāju saskarnes bibliotēku ar jaunām vadīklām, dodot plašākas iespējas variēt lietotāju saskarne;
- Izveidot iespēju personalizēt arī citus programmatūras logus (pašreiz ir iespējams pielāgot tikai programmatūras galveno logu).

IZMANTOTĀ LITERATŪRA

1. **Datorikas institūts DIVI.** FIBU risinājums. [Tiešsaiste] 2011. gada 12. maijs. [Citēts: 2011. gada 14. maijs.] http://di.lv/risinajumi/fibu-risinajums.

2. **Google.** *Google Docs.* [Tiešsaiste] Google. [Citēts: 2011. gada 3. aprīlis.] docs.google.com.

3. —. *Google Maps.* [Tiešsaiste] [Citēts: 2011. gada 3. aprīlis.] http://maps.google.com/.

4. —. Themes. *Google code.* [Tiešsaiste] Google. [Citēts: 2011. gada 9. aprīlis.] http://code.google.com/chrome/extensions/themes.html.

5. **W3Techs.** Usage of content management systems for websites. *Web Technology Surveys.* [Tiešsaiste] 2010. gada 13. septembris. [Citēts: 2011. gada 10. aprīlis.]

6. **Wordpress.** Using Themes. *Wordpress documentation.* [Tiešsaiste] [Citēts: 2011. gada 10. aprīlis.] http://codex.wordpress.org/Using_Themes.

7. —. Free Themes Directory. *Extend Wordpress.* [Tiešsaiste] [Citēts: 2011. gada 10. aprīlis.]

8. **Artisteer.** Artisteer - Automated Web Designer. [Tiešsaiste] [Citēts: 2010. gada 10. aprīlis.] http://www.artisteer.com/.

9. **Wikipedia.** User interface markup language. [Tiešsaiste] [Citēts: 2011. gada 10. aprīlis.] http://en.wikipedia.org/wiki/User_interface_markup_language.

10. —. Wasabi (software). [Tiešsaiste] [Citēts: 2011. gada 27. aprīlis.] http://en.wikipedia.org/wiki/Wasabi_(software).

11. **UIML.org.** Home of the User Interface Markup Language. *UIML.* [Tiešsaiste] [Citēts: 2011. gada 17. aprīlis.] http://www.uiml.org/.

12. **Microsoft.** XAML Overview (WPF). *MSDN.* [Tiešsaiste] [Citēts: 2011. gada 17. aprīlis.] http://msdn.microsoft.com/en-us/library/ms752059.aspx.

13. **Adobe.** MXML. [Tiešsaiste] [Citēts: 2011. gada 17. aprīlis.] http://learn.adobe.com/wiki/display/Flex/MXML.

14. **Phanouriou, Constantinos.** *UIML: A Device-Independent User Interface Markup.* Blacksburg, Virginia : bez nos., 2000.

15. **UIML.org.** UIML tools. *UIML.* [Tiešsaiste] [Citēts: 2011. gada 17. aprīlis.] http://www.uiml.org/tools/index.htm.

16. **Luyten, Kris.** Uiml.Net: a Uiml renderer for .Net. [Tiešsaiste] [Citēts: 2011. gada 7. maijs.] http://research.edm.uhasselt.be/~kris/projects/uiml.net/.

17. **Université catholique de Louvain.** *UsiXML. USer Interface eXtensible Markup Language.* 2004.

18. **Laszlo.** *OpenLaszlo.* [Tiešsaiste] 2005. gada. [Citēts: 2011. gada 17. aprīlis.] http://www.openlaszlo.com/.

19. **Mozilla Developer Network.** XUL. *MDN Doc center.* [Tiešsaiste] 2011. gada 1. aprīlis. [Citēts: 2011. gada 17. aprīlis.] https://developer.mozilla.org/En/XUL.

20. **The Apache Software Foundation.** WTKX Primer. *ApachePivot. Rich internet applications (RIA) in Java.* [Tiešsaiste] [Citēts: 2011. gada 17. aprīlis.]

21. **Prototype Core Team.** *Prototype framework.* [Tiešsaiste] [Citēts: 2011. gada 14. maijs.] http://www.prototypejs.org/.

22. **Limbourg, Quentin un Vanderdonckt, Jean.** *UsiXML: a user interface description language sup-porting multiple levels of independence.* Louvain-la-Neuve : Universite catholique de Louvain, School of Management (IAG), 2004.

23. **Turoff, Murray.** *Personalization in user interface.* 2001.

24. **Wikipedia.** Extensible Application Markup Language. [Tiešsaiste] 2011. gada 3. aprīlis. [Citēts: 2011. gada 21. aprīlis.] http://en.wikipedia.org/wiki/Xaml.

PIELIKUMI

1. pielikums. UIML lietotāju saskarnes apraksta piemērs

```
<?xml version="1.0"?>
<!-- <!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 2.0 Draft//EN" "UIML3_0a.dtd"> -->
<uiml>
    <interface>
        <structure>
        <part id="container" class="Container">
            <part id="calendar" class="Calendar"/>
        </part>
        </structure>
        <style>
            <property part-name="container" name="size">300,300</property>
            <property part-name="calendar" name="position">20,20</property>
            <property part-name="calendar" name="size">100,150</property>
            <property part-name="calendar" name="label">Simple Calendar</property>
            <!-- dates are specified as month/day/year -->
            <property part-name="calendar" name="minDate">01/01/2004</property>
            <property part-name="calendar" name="maxDate">12/31/2004</property>
        </style>
    </interface>
    <peers>
        <presentation base="swf-1.1.uiml"/>
    </peers>
</uiml>
```

2. pielikums. Lietotāju saskarnes apraksta piemērs

Personas apraksts serializēts JSON formātā:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address":
    {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber":
    [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

Personas apraksts serializēts XML formātā:

```
<person>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <age>25</age>
    <address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
    </address>
    <phoneNumber type="home">212 555-1234</phoneNumber>
    <phoneNumber type="fax">646 555-4567</phoneNumber>
</person>
```

3. pielikums. Lietotāju saskarnes apraksta piemērs

```
{
    "settings" : {},
    "layout" :
    [
        {
            "type" : "splitContainer",
            "id" : "mainContainer",
            "anchor" : ["top", "bottom", "left", "right"],
            "style" :
            {
                "top" : 0,
                "left" : 0,
                "right" : 0,
                "bottom" : 0
            },
            "containers" :
            [
                {
                    "type" : "container",
                    "anchor" : ["left", "bottom", "top", "right"],
                    "style" :
                    {
                        "top" : 0,
                        "left" : 0,
                        "width" : 0,
                        "bottom" : 0
                    },
                    "childs" :
                    [
                        {
                            "type" : "control",
                            "control" : "button",
                            "id" : "btnLoad",
                            "options" :
                            {
                                "label" : "Ielādēt"
                            }
                        }
                    ]
                },
                {
                    "type" : "container",
                    "anchor" : ["left", "bottom", "top", "right"],
                    "style" :
                    {
                        "top" : 0,
                        "left" : 0,
```

```
                        "right" : 0,
                        "bottom" : 0
                    },
                    "childs" :
                    [
                        {
                            "type" : "control",
                            "control" : "documentBox",
                            "id" : "documentBox"
                        }
                    ]
                }
            ]
        }
    ],
    "triggers" :
    [
        {
            "trigger" : "btnLoad:click",
            "handler" : "documentBox:load",
            "params" : [{"BLANK_ID" : 40177}]
        }
    ]
}
```

DOKUMENTĀRĀ LAPA

Bakalaura darbs

Informācijas sistēmu saskarnes personalizācija

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

Autors: _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augšminēto bakalaura darbu un atzīstu to par **piemērotu/nepiemērotu** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu bakalaura studiju programmas gala pārbaudījuma komisijas sēdē.

Darba vadītājs(-ja): _____
(Vadītāja paraksts)

Darbs iesniegts Datorikas fakultātē _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.
Metodiķe: _____.
(Metodiķes paraksts)

Recenzents: _____

Darbs aizstāvēts bakalaura darbu gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)