LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DOMĒNA SPECIFISKĀS VALODAS REDAKTORA IZSTRĀDE

KVALIFIKĀCIJAS DARBS

Autors: Ralfs Cimermanis
Studenta apliecības Nr.: rc17006
Darba vadītājs: M.dat. Valdis Vizulis

RĪGA 2019

ANOTĀCIJA

Kvalifikācijas darba "Domēna specifiskās valodas redaktora izstrāde" ietvaros tika veikta jaunas programmatūras izstrāde, kas aizstātu apdrošināšanas platformas Bamboo eSales rīka redaktoru. eSales ir rīku kopums, kuru galvenā funkcija ir apdrošināšanas tīmekļa vietņu satura pārvaldīšana un attēlošana platformas lietotājiem. eSales redaktora galvenā funkcija ir izstrādātājiem viegli veidot un rediģēt dinamiski veidotas eSales tīmekļa vietnes. Jaunas programmatūras izstrādes mērķis ir aizstāt eSales redaktoru, kurš ir veidots uz vecām tehnoloģijām, un galveno kārt uzlabot eSales redaktora efektivitāti.

Atslēgvārdi: Electron, TypeScript, React, spējā izstrāde, XML

ABSTRACT

Within the framework of the qualification work “Domain-specific language editor development”, a new software development was carried out to replace old insurance platforms bamboo eSales tool editor. eSales is a set of tools whose primary function is managing and displaying the contents of insurance websites for the users of the platform. eSales editors’ main function is for developers to easily create and edit dynamically designed eSales websites. The goal of the new software development is to replace the old eSales editor, built on old technologies, and to improve the efficiency of the eSales editor in the first place.

Keywords: Electron, TypeScript, React, Agile Development, XML

SATURS

IEVADS

Informāciju tehnoloģiju uzņēmums SIA "Sapiens Software Solutions" strādā ar apdrošināšanas jomā, kā rezultātā tiek izstrādāta tiešsaistes apdrošināšanas platforma Bamboo, kas ir dažādu rīku un risinājumu komplekts, ar kuru palīdzību tiek nodrošināta apdrošināšanas polišu tirdzniecība tiešsaistē. Tādēļ, Bamboo platformas izstrādes rīku darbības efektivitāte ir būtiski svarīga, jo apgrūtināta lietojamība un ātrdarbība tieši ietekmē produkta izstrādi un uzturēšanu.

Bamboo projektā ietilpst vairākas sistēmas, kur viena no sistēmām ir eSales. eSales ir lietotāju saskarne, kas nodrošina lietotāju mijiedarbību ar sistēmu. Saskarnes būtība ir ļaut lietotājiem aizpildīt datu modeļus, kas pēc tam tiek nodoti uz citu Bamboo sistēmu eRate, kura cieši strādā kopā ar eSales. eRate ir sistēma, kas nodrošina datu apstrādi un klientu prēmijas aprēķināšanu. Bez saskarnes, eSales sastāv arī no eSales redaktora. eSales domēna specifiskās valodas redaktors ir tīmekļa lietotne, kas programmētājiem ļauj viegli veidot un rediģēt dinamiski veidotas tīmekļa vietnes. Veiktās izmaiņas eSales redaktorā tiek glabātas XML datnēs, kur šo datņu nolasīšanu un atveidošanu tīmekļa vietnēs nodrošina vēl viena eSales sistēmas daļa – eSales runtime.

Šobrīd eSales redaktors ir pieejams tikai tīmekļa lietotnes formā un tam ir vairāki trūkumi - apgrūtināta lietojamība, neapmierinoša ātrdarbība un sarežģīti strukturēts kods, kas apgrūtina koda lasāmību un uzlabošanu. Lai risinātu šīs problēmas, uzlabotu XML failu saglabāšanu un ieviestu XML validēšanu, un nodrošinātu labāku sistēmas lietojamību, autoram, kā kvalifikācijas darba mērķis, tika uzstādīts aizvietot eSales sistēmas tīmekļa lietotnes risinājumu ar datora lietojumprogrammu, izstrādājot to izmantojot tehnoloģijas kā Electron, TypeScript un React, kas nākotnē nodrošinātu iespēju, vajadzības gadījumā, viegli pārnest no datora lietojumprogrammas uz tīmekļa lietotnes risinājumu.

Problēmas, kuras tiks risinātas darbā:

- Jauna eSales redaktora izveide, kas aizstātu pašreizējo tīmekļa lapas risinājumu
- Koda tīrīšana un uzlabošana, kas nodrošinātu vieglāku kodu lasīšanu un spēju veikt uzlabojumus.
- Redaktora ātrdarbības un lietojamības uzlabošana nodrošinātu ātrāku izstrādes procesu.

Izstrādes process tiks veikts pēc spējās izstrādes metodoloģijas, kur lietotāju stāsti tiks sadalīti 2 nedēļu garās iterācijās.

VĀRDNĪCA

1.tabula

Vārdnīcas termini un skaidrojumi

Jēdziens	**Skaidrojums**
Electronjs	Rīks multi-platformu lietotņu veidošanai ar HTML, CSS un JavaScript valodām.
CDATA	XML dokumenta daļa, kas satur vispārīga rakstura simbolus. CDATA sekcija netiek uztverta, kā daļā no iezīmēšanas valodas.
Datu plūsmu diagramma (DPD)	Modelēšanas rīks, kas ļauj parādīt sistēmu kā funkcionālu procesu tīklu.
ER diagrammas (ang. *Entity–relationship model*)	Datu bāzu modelēšanas metode, kura tiek izmantota, lai izveidotu datu bāzes shēmu vai datu modeli un izprastu veidojamās datu bāzes uzbūvi un pamatprincipus. Diagrammas parasti izmanto relāciju datu bāzes izstrādē.
eSales cilpa(ang. *loop*)	eSales struktūras elements, kas apraksta kādu darījumprocesu plūsmu no sākuma līdz beigām.
Lookup	eSales krītošo izvēļņu elementu vērtības
Programmatūras prasību specifikācija (PPS)	Specifikācija atsevišķam programmatūras produktam, programmai vai programmu kopai, kas izpilda noteiktas funkcijas.
Programmatūras projektējuma apraksts (PPA)	Programmatūras sistēmas attēlojums, kas tiek radīts, lai atvieglotu analīzi, plānošanu, implementēšanu un lēmumu pieņemšanu. Tas ir programmatūras sistēmas uzmetums vai modelis.
Redmine	Atvērtā pirmkoda projektu pārvaldības sistēma.
SCRUM	Spējās izstrādes ietvars, kas ir pielāgots lieliem un sarežģītiem projektiem.
Git	Versiju kontroles rīks.
TSLint	Analīzes rīks, kas pārbauda TypeScript koda lasāmību, uzturamību un koda kļūdas.
Prettier	Koda formatēšanas rīks, kas padara kodu lasāmāku.

1. PROGRAMMATŪRAS PRASĪBU SPECIFICKĀCIJA

1.1 Ievads

1.1.1 Nolūks

Šis dokuments ir programmatūras prasību specifikācija (PPS). PPS nolūks ir aprakstīt un apkopot sistēmas funkcionālas (lietotāju stāsti) un nefunkcionālās prasības. Dokuments ir paredzēts sistēmas izstrādātājiem, kā tehniskās specifikācijas un apraksta palīgmateriāls.

1.1.2 Darbības sfēra

"eSales Redaktors", turpmāk eSales redaktors, ir domēna specifiskās valodas redaktors, ar kura palīdzību tiek veidotas darījumprocesu plūsmas, un kuru izmanto *eSales runtime*, kas nodrošina dinamisku tīmekļu vientes atveidošanu. Redaktora mērķis ir aizstāt redaktoru tīmekļa vietnē, uzlabojot redaktora ātrdarbību un lietojamību. Redaktoru izmantos apdrošināšanas platformas un eSales sistēmas izstrādātāji.

1.1.3 Saistība ar citiem dokumentiem

Šis dokuments – PPS ir saistīts ar programmatūras projektējuma aprakstu (PPA) un testēšanas dokumentāciju.

1.1.4 Pārskats

Dokuments sastāv no 4 daļām.

Pirmā daļa satur ievadu, kas sniedz vispārīgu informāciju par dokumenta nolūku, izstrādājamā produkta mērķi, kā arī tā saistību ar citiem dokumentiem.

Otrajā daļā ietverts vispārējs produkta apraksts – produkta perspektīva saistībā ar citiem saistītiem produktiem vai projektiem, kopsavilkums par sistēmas funkcijām, produkta lietotāja raksturiezīmes, kuras iespaido specifiskās prasības, kā arī pieņēmumi un atkarības, kas sastāv no visiem tiem faktoriem, kas iespaido PPS prasības.

Trešā daļa sastāv no konkrētām funkcionālajām prasībām, kas izklāstītas lietotāju stāstu formā.

Ceturtā daļa ietver sistēmas nefunkcionālās prasības.

1.2 Vispārējais apraksts

1.2.1 Produkta perspektīva

Darba ietvaros izstrādātā lietotne ir patstāvīga, kura strādā jau ar esošiem eSales datiem sistēmā, pildot funkcijas kā jaunu datu pievienošanu, dzēšanu un kopēšanu. Lietotne ir pieejama tikai apdrošināšanas sistēmas izstrādātājiem.

1.2.2 Sistēmas lietotāji

Lietotne nodrošina vienu lietotāju grupu:

- "eSales izstrādātājs" veic visas darbības saistībā ar sistēmu (dzēš, pievieno, kopē un rediģē datus)

1.2.3 Produkta funkcijas

- Visu XML datņu asinhrona ielāde
- Lietotnes datu attēlošana kokos
- Koka elementu pievienošana, kopēšana, dzēšana, atvēršana, un pārvietošana
- Koka elementu informācijas attēlošana un rediģēšana
- Koka elementu un to satura meklēšana

1.2.4 Lietotāju raksturiezīmes

eSales redaktoru lietos eSales izstrādātāji, kas nozīmē, ka lietotājiem ir jābūt zināšanām par eSales darbības principiem, komandrindas izmantošanu rīka uzbūvei un darbībai ar datoru lietojumprogrammām.

1.2.5 Pieņēmumi un atkarības

Tiek pieņemts, ka lietotājiem, kuri vēlas strādāt ar lietotni, ir pieejami datori, kuri atbalsta Electronjs tehnoloģiju, un piekļuvi uzņēmuma iekšējam tīklam, lai piekļūtu sistēmas datiem un eSales rīkiem, kā arī būtu spējīgi veikt izmaiņas datos.

1.3 Funkcionālās prasības

Šajā nodaļā ir aprakstītas funkcionālas prasības lietotājstāstu formā pēc spējās izstrādes metodoloģijas. Lietotāj stāsti tika sadalīti trīs fāzēs, kur darba gaitā tika izstrādātas divas. Katrai prasībai ir piešķirts unikāls identifikators, kas ņemts no apdrošināšanas sistēmas Redmine – atvērtā pirmkoda projektu pārvaldības rīks. Prasības tiek attēlotas tabulās ar sadaļām: stāsta numurs, stāsta apraksts, akceptēšanas kritēriji, identifikators. Papildus tiek pievienota sadaļa validācijas paziņojumi, stāstiem, kuru funkcionalitāte atgriež kļūdu paziņojumus.

1.3.1 Programmatūras izstrādes pirmā fāze

Pirmās fāzes ietvaros tiks izstrādātas galvenās pamatfunkcijas, lai būtu iespējams sākt izmantot redaktoru lasīšanas režīmā un iegūt pirmās atsauksmes no programmētājiem par ieteikumiem un turpmāk vajadzīgajiem uzlabojumiem.

1.3.1.1 Izstrādāt pamatfunkcijas (XML datņu ielādi un saglabāšanu)

Stāsta numurs
#97731
Stāsts
Kā *eSales* izstrādātājs **Vēlos** Izstrādāt XML datņu ielādi datu modeļos un to saglabāšanu atpakaļ XML datnēs. **Lai** nodrošinātu redaktoram pieeju pie datiem atmiņā un spēju saglabāt izmaiņas.
Akceptēšanas kritēriji
• Asinhrona datņu ielāde klasēs • Prioritizēta Loop.xml datņu ielāde • Saglabājot atpakaļ XML datnēs nav izmaiņas to struktūrā un formātā • Pareiza CDATA saglabāšana
Validācijas paziņojumi
• Ja ielādes direktorija neeksistē tiek izdots kļūdas paziņojums - “Can’t find directory *Direktorija*” • Ja saglabāšanas direktorija neeksistē tiek izdots kļūdas paziņojums - “Can’t find directory *Direktorija*”

1.3.1.2 Izstrādāt datu izmaiņu saglabāšanu

Stāsta numurs
#97732
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** redaktors saglabā izmaiņas, ja tiek veikta informācijas atjaunošana **Lai** izstrādātāji varētu saglabāt veiktās izmaiņas
Akceptēšanas kritēriji
• Izmaiņas tiek saglabātas modeļos automātiski • Tiek pārbaudīts, vai formas teksta logos tiek rakstīta pareiza tipa informācija. • Ieviests identifikators, kas norāda, vai ir veiktas izmaiņas kādā objektā • Redaktors saglabā no jauna tikai tās XML datnes, kurās ir veiktas izmaiņas
Validācijas paziņojumi
• Skaitļu ievades formā netiek ievadīts skaitlis – "*Formas nosaukums* only accepts numbers"

1.3.1.3 Izstrādāt JSON datņu maketus priekš kontroļu tipiem, atribūtu uzmeklēšanas un unikālu ID vērtībām

Stāsta numurs
#97734
Stāsts
Kā *eSales* izstrādātājs **Vēlos** izveidot JSON datņu maketus priekš *Question* kontroļu tipiem, atribūtu *Lookup* vērtībām un unikāliem ID **Lai** redaktors varētu strādāt lasīšanas režīmā bez tīmekļa pakalpojumiem
Akceptēšanas kritēriji
• Datņu struktūra atbilst gaidāmajai informācijai no tīmekļa pakalpojumiem • Atsevišķs makets unikālu ID vērtībām un apvienots kontroļu tipu un atribūtu uzmeklēšanas makets • Redaktors ielasa datnes atmiņā pie programmas uzsākšanas
Validācijas paziņojumi

• Ja datne neeksistē tiek izdots kļūdas paziņojums - “Can’t find file *Datnes nosaukums* in *Direktorija*” • Ja failu direktorija neeksistē tiek izdots kļūdas paziņojums - “Can’t find directory *Direktorija*”

1.3.1.4 Izstrādāt pamata lietotāja saskarni

Stāsta numurs
#97735
Stāsts
Kā *eSales* izstrādātājs **Vēlos** izveidot viegli lietojamu lietotāju saskarni, kopā ar koka attēlošanu **Lai** ir iespējams sāk izmantot redaktoru lasīšanas režīmā
Akceptēšanas kritēriji
• Katrs produkts tiek attēlots koka formā • Izvēloties produktu, tā elementi tiek attēloti jaunā kokā • Izvēloties kādu produkta elementu, tā dati tiek attēloti tam attiecīgā formā. • Katra elementa tipam ir dažādi funkcionējošas pogas • Ir meklēšanas josla • Strādā lasīšanas režīmā

1.3.1.5 Izveidot katram eSales datu elementu tipiem attiecīgu formu

Stāsta numurs
#99252
Stāsts
Kā *eSales* izstrādātājs **Vēlos** katram elementa tipam (Loop, Page, Question, DisplayCondition) izveidot savu rediģēšanas skatu. **Lai** izstrādātāji būtu spējīgi rediģēt koka elementu atribūtu vērtības
Akceptēšanas kritēriji
• Katram elementam ir izveidota sava komponente rediģēšanas skata attēlošanai. • Loop, Page, DisplayCondition formu struktūras tiek ielādētas no redaktora.

- Question skata lauki tiek attēloti pēc Question tipa veida, balstoties pēc eSalesEditor definīcijām
- Question skata lauki tiek sagrupēti kategorijās
- Katras formas laukam pielikts klāt identifikators, ja tā vērtība neatbilst noklusētajai vērtībai.

1.3.2 Programmatūras izstrādes otrā fāze

Otrās fāzes nolūks ir uzlabot programmatūras pirmās fāzes funkcionalitāti ar svarīgām rediģēšanas funkcijām, piemēram, kā elementu meklēšanu, jaunu elementu pievienošanu un kopēšanu. Otrā fāzes izstrādes laikā arī tiek ņemti vērā programmētāju ieteikumi un atsauksmes redaktora uzlabošanai.

1.3.2.1 Izstrādāt meklēšanu

Stāsta numurs
#97739
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** var meklēt elementus vērtības, kas satur meklēšanas frāzi **Lai** atvieglotu koku elementu meklēšanu
Akceptēšanas kritēriji
• Meklēšana tikai viena Loop koka ietvaros • Meklēšana tikai viena produkta koka ietvaros • Meklēšana notiek regulārās izteiksmes režīmā • Tiek izveidots jauns koks, kura vērtības satur meklēšanas frāzi • Ievadot simbolu tiek uzreiz izpildīta meklēšana

1.3.2.2 Izstrādāt koka elementu kopēšanu un pievienošanu

Stāsta numurs
#97742
Stāsts
Kā *eSales* izstrādātājs **Vēlos** katram elementa tipam (Loop, Page, Question, DisplayCondition) pievienot iespēju izveidot jaunu elementu un kopēt to koka ietvaros. **Lai** būtu iespējams paplašināt esošo sistēmu
Akceptēšanas kritēriji
• Elementa kopēšana ir pieejama visiem elementiem • Jauna Question elemenetu var pievienot tikai Page elementiem vai Question elementiem ar GroupControl kontroles tipu • Jaunu Page elementu var pievienot tikai Loop ietvaros • Izveidojot jaunu elementu vai kopējot, automātiski tiek atjaunots koks • Jaunais elements tiek automātiski atvērts rediģēšanai • Izveidojot jaunu vai kopējot elementu, elementa un tā visu bērnu ID tiek nomainīti pret jauniem negatīviem ID

1.3.2.3 Koka elementu vecāku maiņa

Stāsta numurs
#97744
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** Page, Question un DisplayCondition elementus ir iespējams mainīto to atrašanās vietu **Lai** jaunus un kopētus elementus varētu novietotiem tiem paredzētā vietā
Akceptēšanas kritēriji
• Katram elementam ir iespēja to pārvietot zem cita elementa • Jebkuru Question elementu var pārvietot tikai zem cita Question elementa ar GroupControl kontroles tipu • Question elementus ar GroupControl kontroles tipu var pārvietot tieši zem Page elementa.

• Pārvietojot elementu, visi tā bērni pārvietojās līdzi • Pārvietotā elementa references ID tiek nomainīts uz jaunā vecāka ID • Page elementus var pārvietot tikai starp Loop elementiem • DisplayCondition elementus var pārvietot tikai starp Loop elementiem • Pēc pārvietošanas koks, kurā tika veiktas izmaiņas, tiek pārlādēts

1.3.2.4 Question elementu kontroles tipa maiņa

Stāsta numurs
#97745
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** Question elementiem ir iespējams nomainīt kontroļa tipu **Lai** katru reizi nebūtu jātaisa jauns elements
Akceptēšanas kritēriji
• Question elementiem, kuru kontroles tips ir GroupControl un elementam ir bērni, maiņa netiek piedāvāta. • Nomainot tipu, redaktors automātiski atjauno labošanas formu atbilstoši kontroles tipa definīcijai. • Aizpildīto lauku, kas sakrīt gan vecā, gan jaunā kontroles definīcijās, vērtības tiek saglabātas

1.3.2.5 Izstrādāt saistītu Loop elementu attēlošanu

Stāsta numurs
#97746
Stāsts
Kā *eSales* izstrādātājs **Vēlos** attēlot iestarpinātus Loop elementu produktu lapu kokā **Lai** nebūtu jāmeklē citos Loop saistītie elementi.
Akceptēšanas kritēriji
• Produktu koka lapu struktūrā tiek parādīti saistītie Loop elementi • Saistītie elementi tiek attēloti citā krāsā

• Saistītajiem elementiem ir pievienots Loop identifikators, no kura viņi nāk. • Saistītos elementus var tikai rediģēt atveros tos no to orģinālās atrašanās vietas

1.3.2.6 Pievienot koka elementu dzēšanu

Stāsta numurs
#97747
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** visiem koka elementiem būtu iespēja tos izdzēst **Lai** kļūdu vai nevajadzīgu elementu gadījumā tos ir iespējams dzēst
Akceptēšanas kritēriji
• Katra elementa dzēšana ir pieejama to rediģēšanas formā. • Pirms dzēšanas tiek pārjautāts, vai elements tiešām jādzēš • Tiek pārlādēts koks elementa dzēšanas gadījumā • Dzēšot elementu tiek arī dzēsti visi tā bērni • Dzēšot DisplayCondition vai Page elementu tiek dzēstas visas references uz to elementu

1.3.2.7 Pievienot koka ikonas katram elementa tipam

Stāsta numurs
#99434
Stāsts
Kā *eSales* izstrādātājs **Vēlos** pievienot elementu ikonas koka struktūrā **Lai** kokā būtu vieglāk identificēt elementus
Akceptēšanas kritēriji
• Katram elementam tipam ir atšķirīga ikona • Question elementiem ar bērniem un bez bērniem ir atšķirīgas ikonas • Question elementiem ir pievienots identifikators, ja tas satur DisplayCondition un, vai atribūts “NegateDisplayConditionOrBusinessRule” ir ar vērtību True.

1.3.2.8 Izstrādāt globālu meklēšanu

Stāsta numurs
#100207
Stāsts
Kā *eSales* izstrādātājs **Vēlos, lai** ir iespējams meklēt visos koka elementu atribūtos pēc nosaukuma un vērtības **Lai** iespējotu meklēšanu viena vai vairāku Loop elementu meklēšanu
Akceptēšanas kritēriji
• Meklēšanas logā var rakstīt simbolu virkni vai regulāro izteiksmi • Ja ievadītā izteiksme nav regulārās izteiksmes formā, tā tiek automātiski pārvērsta • Meklēšana notiek gan pēc nosaukuma, gan pēc elementu atribūtu satura • Meklēšanas rezultāti tiek attēloti atsevišķā logā • Meklēšanas rezultāti tiek grupēti pēc elementu tipa • Meklēšanas rezultāti satur norādi, kas atver elementu kokā un rediģēšanas logā • Meklēšanas rezultāti, kas satur meklēšanas frāzi, tiek iekrāsoti

1.4 Nefunkcionālās prasības

Šajā nodāļā ir aprakstītas izstrādājamās sistēmas nefunkcionālās prasības, kas ietver sevī drošību un uzturamību.

Drošība

- Tikai cilvēki, kuriem ir pieeja pie darba datoriem un eSales redaktora koda, var piekļūt pie redaktora
- Lai redaktors saglabātu jaunos elementus ir jābūt autorizētam uzņēmuma iekšējā tīklā

Uzturamība

- Sistēmas daļas ir izstrādātas pēc objekt orientētās paradigmas
- Sistēmai ir jābūt izstrādātai pa moduļiem, vieglai funkcionalitātes papildināšanai

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1 Ievads

2.1.1 Nolūks

Šis dokuments ir programmatūras projektējuma apraksts (PPA). PPA mērķis ir nodrošināt veiksmīgu sistēmas izstrādi atbilstoši programmatūras prasību specifikācijā izvirzītajām prasībām. Dokuments ir paredzēts sistēmas izstrādātājiem, kā tehniskās specifikācijas un apraksta palīgmateriāls.

2.1.2 Darbības sfēra

"eSales Redaktors" ir lietotne, kuras mērķis ir aizstāt redaktoru tīmekļa vietnē, uzlabojot redaktora ātrdarbību un lietojamību. Redaktoru izmantos apdrošināšanas platformas un eSales rīka izstrādātāji.

2.1.3 Saistība ar citiem dokumentiem

Šis dokuments – PPA ir saistīts ar programmatūras projektējuma specifikāciju (PPS) un testēšanas dokumentāciju.

2.1.4 Pārskats

Dokuments sastāv no 4 daļām.

Pirmā daļa satur ievadu, kas sniedz vispārīgu informāciju par dokumenta nolūku, izstrādājamā produkta mērķi, kā arī saistību ar citiem dokumentiem.

Otrajā daļa satur datu plūsmu diagrammas, kas dod priekštatu kā sistēmas daļas jeb moduļi komunicē savā starpā un kāda veida dati plūst.

Trešā daļa satur datu bāžu projektējumu ar diagrammām konceptuālajā un fiziskajā līmenī, kā arī tabulu aprakstu.

Ceturtā daļa veltīta sistēmas skatu projektējuma aprakstam.

2.2 Datu plūsmu diagrammas

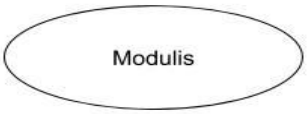
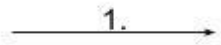
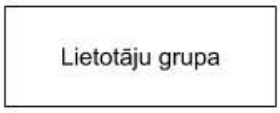
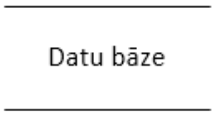
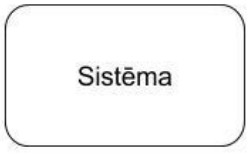
Datu plūsmu diagrammās ir attēlotas tās sistēmas daļas, kuras tika izstrādātas vai modificētas kvalifikācijas darba ietvaros. Moduļos netiek atspoguļota visas sistēmas datu plūsma.

Datu plūsmas starp moduļiem ir numurētas ar identifikatoriem, kuru atšifrējums ir atrodams zem katras attiecīgās datu plūsmu diagrammas.

Datu plūsmas diagrammu apzīmējumi:

2.1. tabula

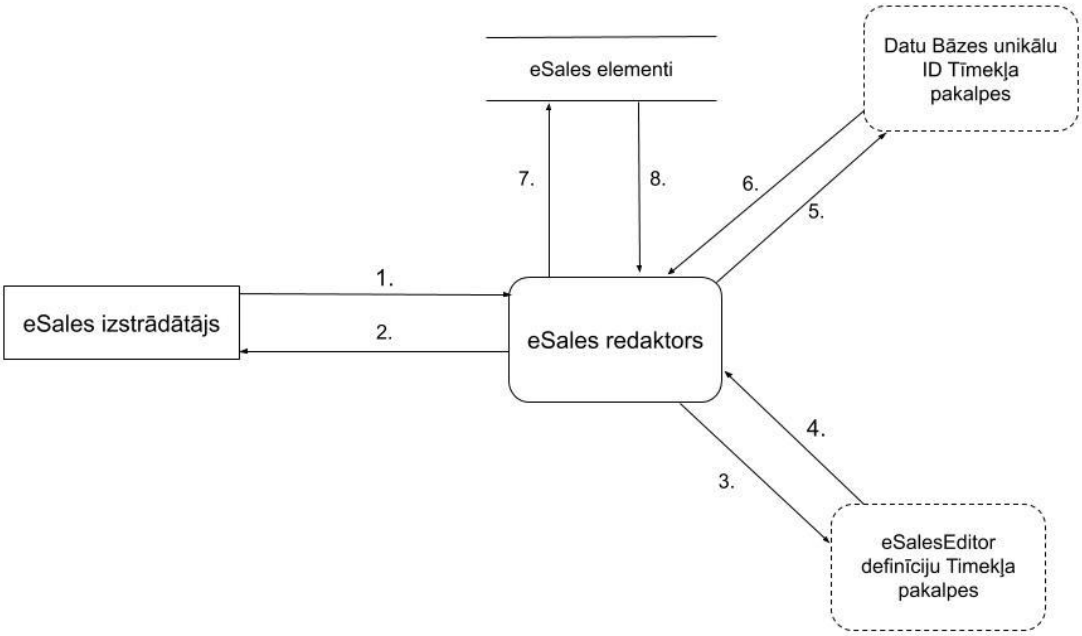
Datu plūsmu diagrammu apzīmējumi

Apzīmējums	**Paskaidrojums**
	Sistēmas modulis (1. līmeņa DPD)
	Datu plūsma
	Lietotāju grupa
	Datu bāze
	Sistēma
	Ārēja sistēma

	Funkcija (2. līmeņa DPD)

2.2.1 0. Līmeņa DPD

0.Līmeņa DPD paskaidro visi lietotājstāsti.

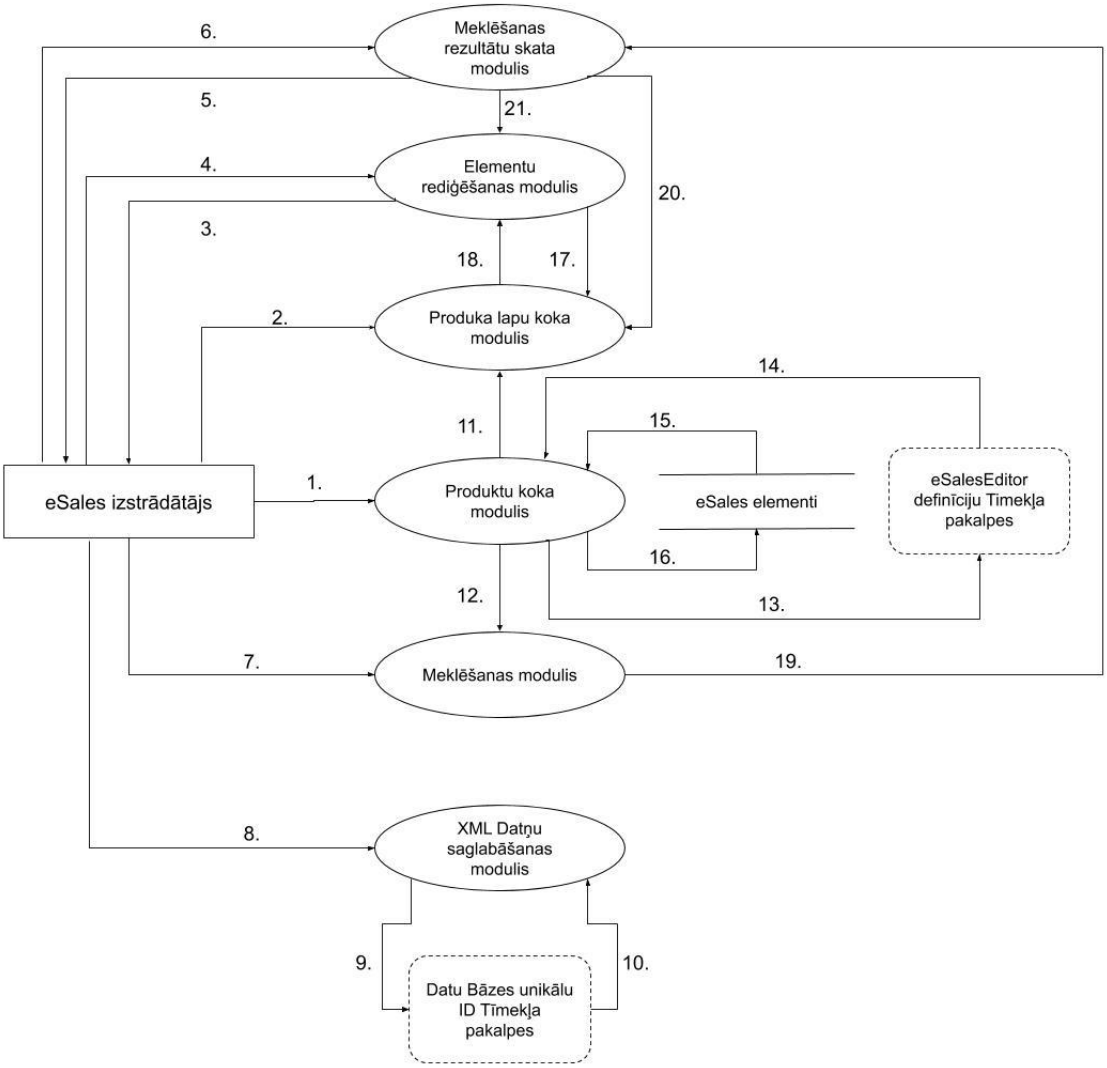


2.1. att. ***datu plūsmu diagrammas 0. līmenis***

1. eSales izstrādātājs pievieno, atjauno un labo eSales elementu saturu
2. eSakes izstrādātājs saņem esošo eSales elementu saturu
3. eSales redaktors pieprasa eSales elementu definīciju aprakstu no eSalesEditor definīciju Tīmekļa pakalpes
4. eSalesEditor definīciju saturs ar atribūtu uzmeklēšanas vērtībām
5. Precīzs skaits jaunu ID vērtību priekš katra jauni veidota elementa saglabāšanas brīdī
6. Unikāli ID
7. Pieprasījums pēc datu modeļa
8. eSales datu modelis

2.2.2 1. Līmeņa DPD

1.Līmeņa DPD paskaidro visi lietotājstāsti.



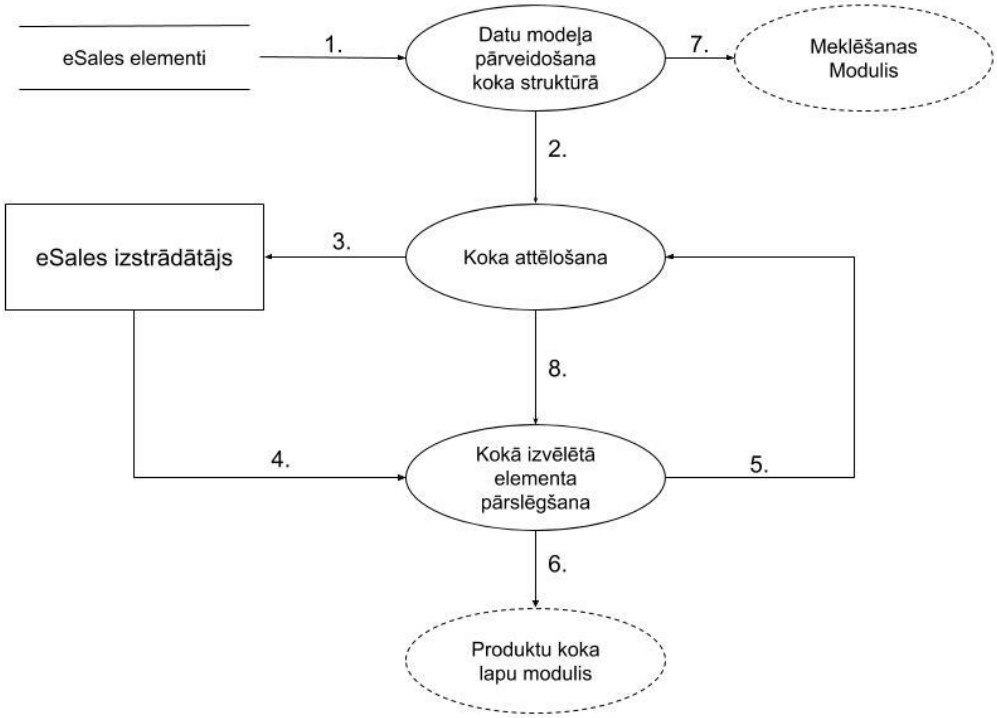
2.2. att. ***datu plūsmu diagrammas 1. līmenis***

1. Izvēlētās produkta cilpas nosaukums
2. Produkta lapu koka elementa ID
3. Informācija par elementa atribūtiem
4. Lietotāja ievadīti datu labojumi
5. Meklēšanas rezultāts
6. Izvēlētā meklēšanas elementa ID
7. Meklēšanas parametrs
8. Glabāšanas pieprasījums

9. Unikālu ID pieprasījums
10. Unikāli elementu ID
11. Produkta cilpas modelis
12. Produkta cilpas modelis
13. Produktu atribūtu definīciju pieprasījums
14. eSales atribūtu definīciju saraksts
15. eSales elementu ielāde
16. Atjaunoti dati
17. Atjaunotie dati
18. Produkta lapu koka elementu atribūti, atribūtu definīcijas
19. Atlasīti meklēšanas rezultāti
20. Elementa ID
21. Elementa ID

2.2.3 2. Līmeņa DPD – Produktu koka modulis

Produkta koka moduli paskaidro lietotājstāsti ar numuriem - #97731, #97734, #97735.

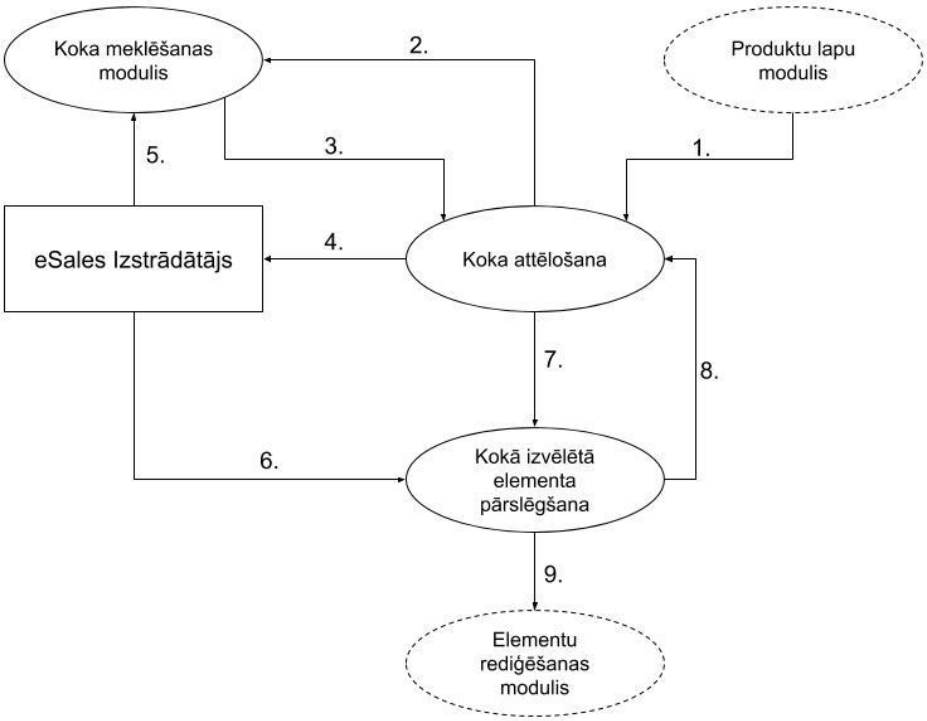


2.3. att. ***datu plūsmu diagrammas 2. līmenis – Produktu koka modulis***

1. eSales elementi
2. Dati koka izveidei
3. Formatēts produktu koks
4. Izvēlētais koka elements
5. Koka elementa ID un atrašanās kokā
6. Izvēlētā produkta cilpas datu modelis
7. Izvēlētā produkta datu modelis
8. Koka elementu ID

2.2.4 2.Līmeņa DPD – Produktu lapu koka modulis

Produkta lapu koka moduli paskaidro lietotājstāsti ar numuriem - #97735, #97739, #99434, #97746

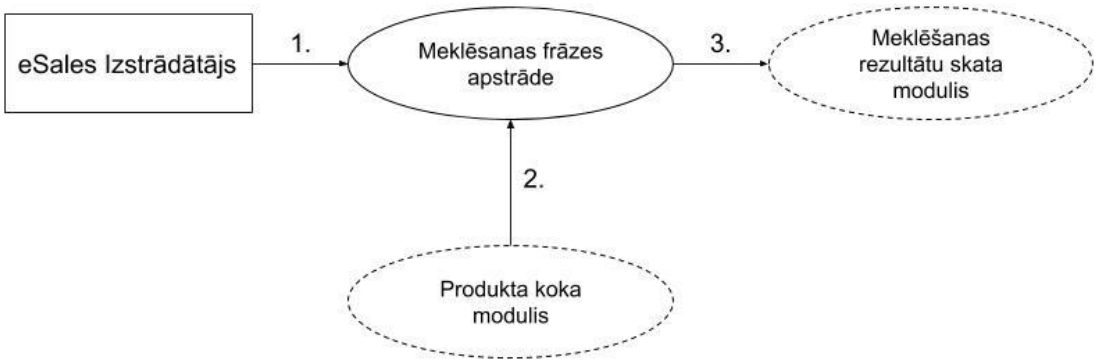


2.4. att. ***datu plūsmu diagrammas 2. līmenis – Produktu lapu koka modulis***

1. Produktu cilpas datu modelis
2. Koka elementu ID un Nosaukumi
3. Elementi, kuri satur meklēšanas frāzi
4. Attēloti produktu cilpas elementi koka struktūrā
5. Meklēšanas frāze
6. Izvēlētā elementa ID
7. Koka elementu ID
8. Koka elementa ID un atrašanās kokā
9. Produkta lapu koka elementu atribūti, atribūtu definīcijas

2.2.5 2. Līmeņa DPD – Meklēšanas modulis

Meklēšanas moduli paskaidro lietotājstāsti ar numuriem - #100207, #97735

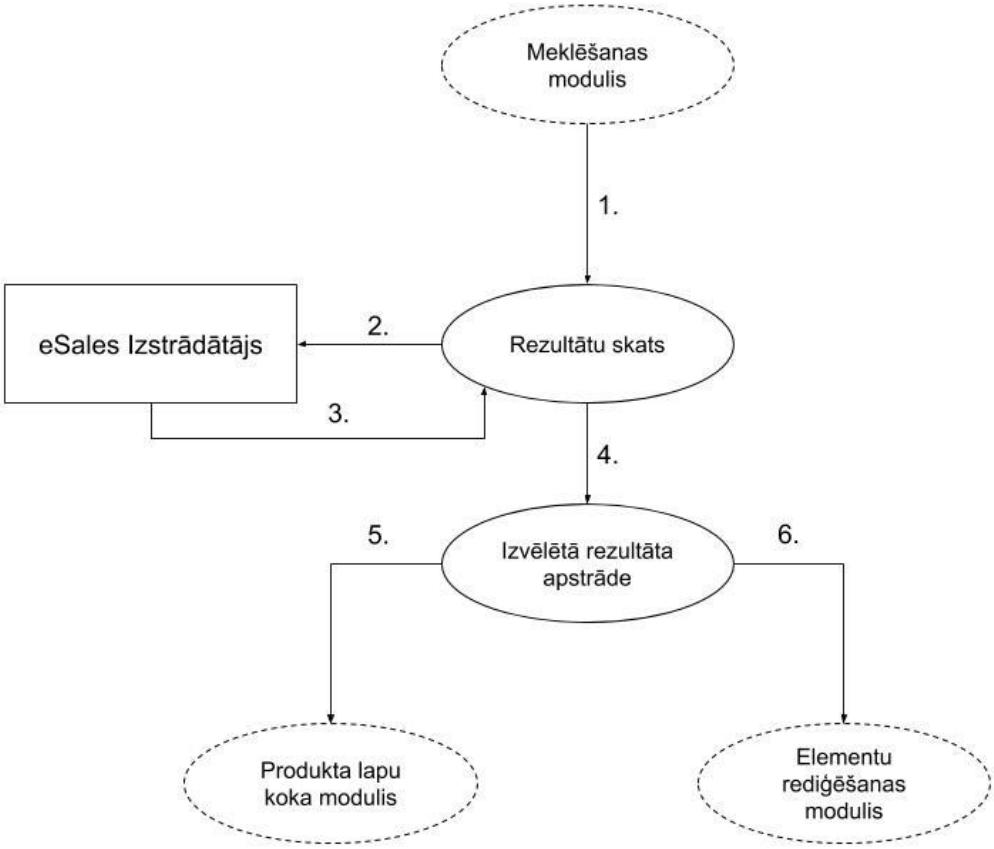


2.5. att. ***datu plūsmu diagrammas 2. līmenis – Meklēšanas modulis***

1. Meklēšanas frāze
2. Produkta cilpas modulis
3. Padotā moduļa elementi, kuru atribūtu vērtības sakrīt ar meklēšanas frāzi

2.2.6 2. Līmeņa DPD – Meklēšanas rezultātu skata modulis

Produkta koka moduli paskaidro lietotājstāsti ar numuriem - #100207, #97735

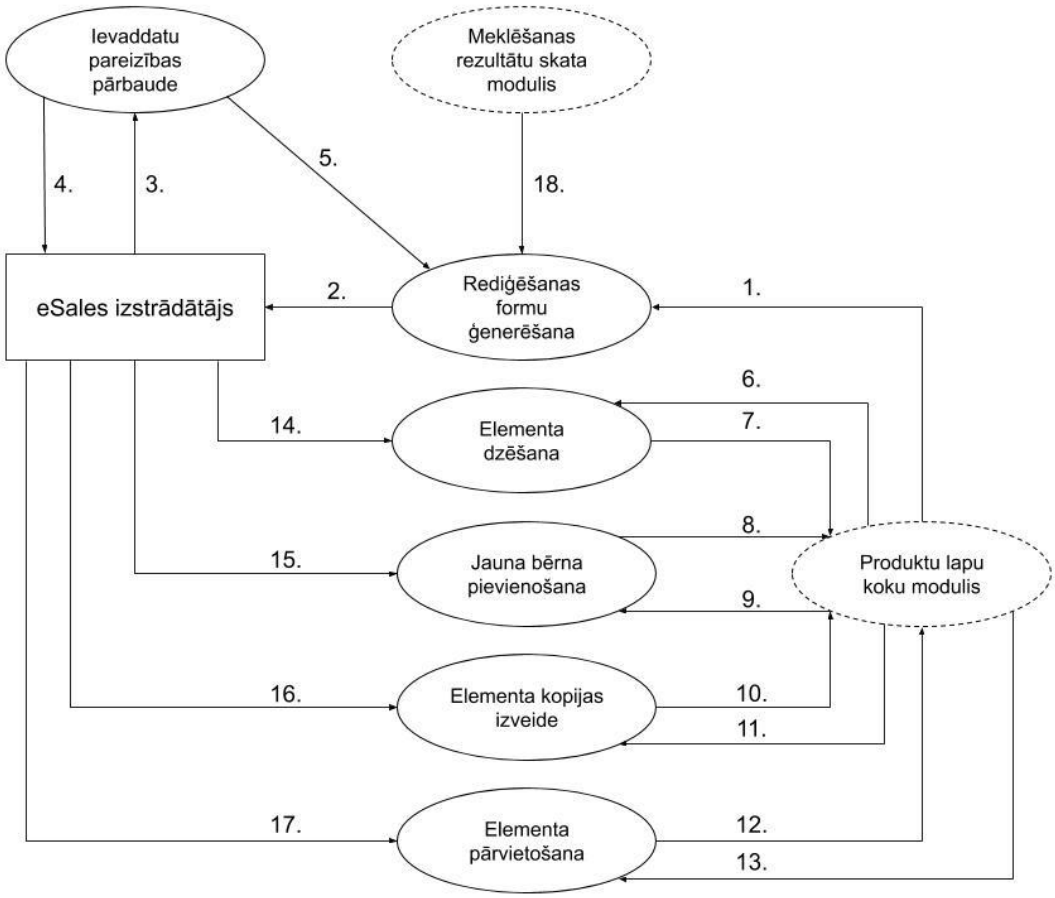


2.6. att. ***datu plūsmu diagrammas 2. līmenis – Meklēšanas rezultātu skata modulis***

1. Izfiltrēti koka elementu atribūti
2. Formatēts meklēšanas rezultāts
3. Izvēlētā rezultāta ID
4. Elementa ID, tips
5. Elementa ID
6. Elementa atribūti

2.2.7 2. Līmeņa DPD – Elementu rediģēšanas modulis

Elementu rediģēšanas moduli paskaidro lietotājstāsti ar numuriem - #97732, #97734, #97735, #99252, #97724, #97744, #97745, #97747



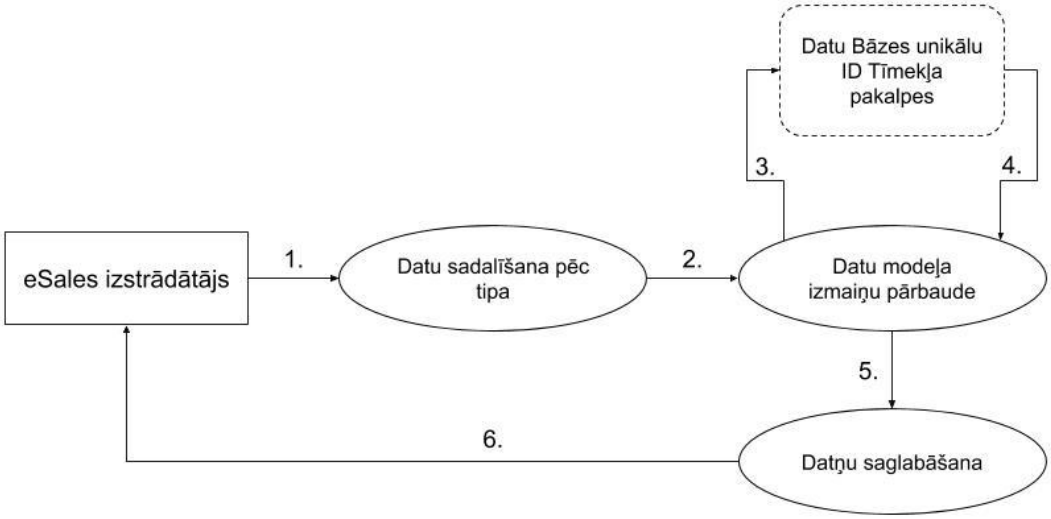
2.7. att. ***datu plūsmu diagrammas 2. līmenis – Elementu rediģēšanas modulis***

1. Produkta lapu koka elementu atribūti, atribūtu definīcijas
2. Sagrupēts rediģēšanas skats
3. Datu labojumi
4. Kļūdas paziņojums
5. Pareizu datu ierakstīšana formā
6. Elementa ID
7. Noņemtā elementa ID
8. Jaunā bērna elementa ID un atrašanās kokā
9. Vecāka elementa ID
10. Kopijas elementa un tā bērnu jaunie ID un atrašanās kokā
11. Kopējamā elementa ID

12. Jaunā vecāka ID
13. Vecā elementa ID un vecāka ID
14. Elementa dzēšanas komanda
15. Jauna bērna pievienošanas komanda
16. Elementa kopijas izveides komanda
17. Elementa pārvietošanas komanda
18. Meklēšanas rezultātu izvēlētā elementa ID un tips

2.2.8 2. Līmeņa DPD – XML datņu izveides modulis

Produkta koka moduli paskaidro lietotājstāsti ar numuriem - #97731



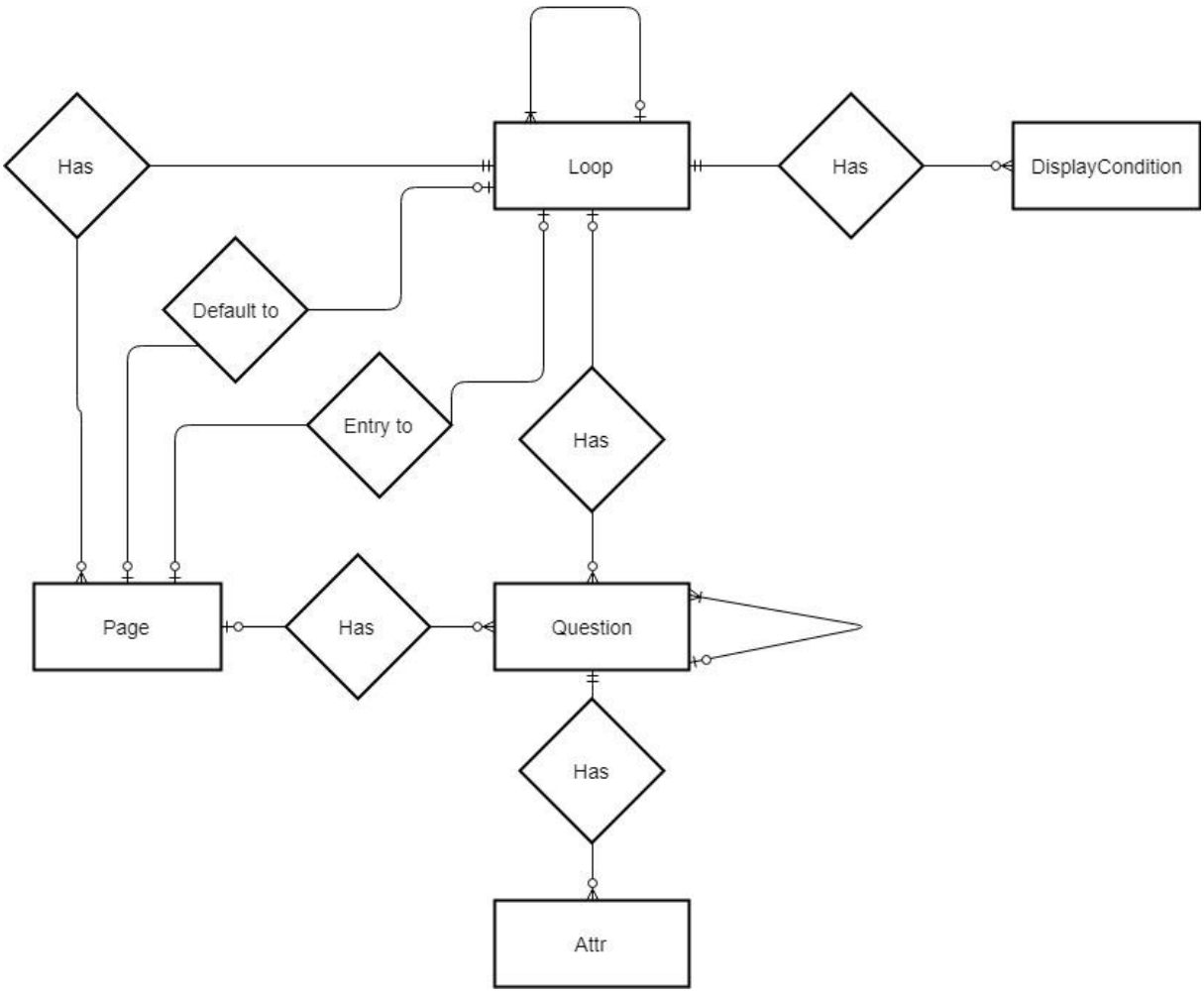
2.8. att. ***datu plūsmu diagrammas 2. līmenis – XML datņu izveides modulis***

1. Saglabāšanas izsaukums
2. Pēc tipiem sagrupēti elementi
3. Jaunu elementu ID pieprasījums
4. Jauni elementu ID
5. Elementi, kuros ir veiktas izmaiņas
6. Saglabāšanas izpildīšanas statuss

2.3 Datubāzes projektējums

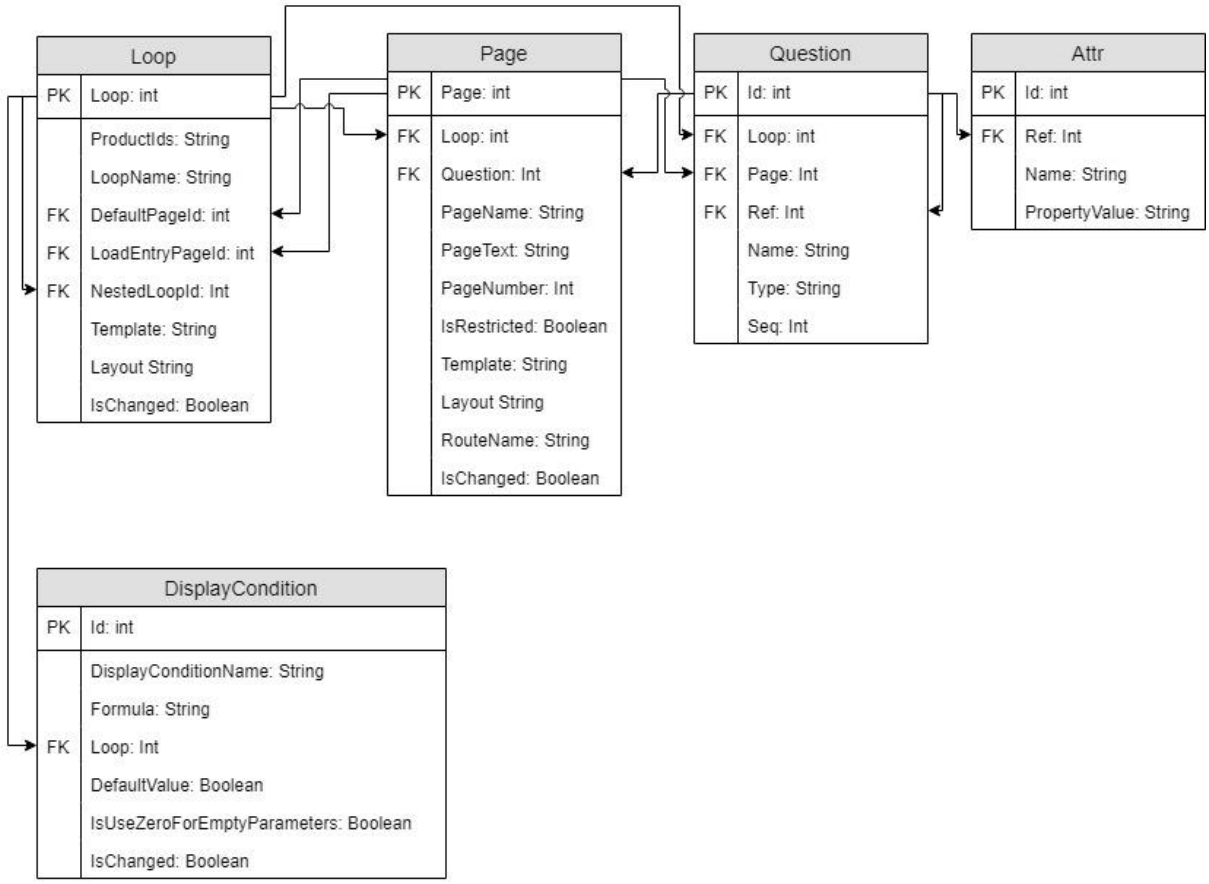
Šajā nodaļā ir aprakstīts eSales modelis, kas tiek izmantots tradicionālās datubāzes vietā, un tā glabāšanas shēma, kura tiek lietota visu eSales datu glabāšanai XML datnēs. Konceptuālais eSales modelis sastāv no piecām entītijām un esošām kardinalitātēm.

2.3.1 Konceptuālais ER eSales modelis



2.9. att. ***konceptuālā eSales modeļa ER diagramma***

2.3.2 Fiziskais ER modelis



2.10. att. ***fiziskais eSales modeļa ER modelis***

2.3.3 eSales datu modeļa tabulu apraksts

eSales datu modeļa tabulu izmantotie apzīmējumi:

- PK – primārā atslēga
- FK – Ārējā atslēga
- Inc – unikāls tabulas elements, kas tabulā automātiski pieaug par 1
- Not Null – Lauka vērtība nedrīkst būt tukša

Tabula "Loop"

Tabulā tiek uzglabāta informācija par eSales modeļa Loop elementu konfigurāciju.

2.2. tabula

"*Loop*" datubāzes tabulas projektējums.

Nosaukums	Tips	Papildus Informācija	Apraksts
Loop	Int	PK, Not Null, Inc	Loop elementa identifikators
ProductIds	String	Not null	Produktu identifikatoru saraksts, kur katrs identifikators ir atdalīts ar semikolu
LoopName	String	Not null	Loop elementa nosaukums
DefaultPageId	Int	FK uz Page	Apdrošināšanas piedāvājuma sākuma lapa
LoadEntryPageId	Int	FK uz Page	Esoša apdrošināšana piedāvājuma lapa
NestedLoopId	Int	FK uz Loop	Papildus Loop identifikators atkārtotai elementu izmantošanai
Template	String		Izklājumu veidnes nosaukums

Layout	String		Izklājumu struktūras nosaukums
IsChanged	Boolean		Pārbaudes vērtība, kas nosaka, vai ir bijušas izmaiņas modelī

Tabula "DisplayCondition"

Tabula satur informāciju par eSales modeļa DisplayCondition elementiem, kuri apraksta nosacījumus par Question elementu attēlošanu.

2.3. tabula

"*DisplayCondition*" datubāzes tabulas projektējums

Nosaukums	Tips	Papildus Informācija	Apraksts
Id	Int	PK, Not Null, Inc	DisplayCondition elementa idnetifikators
DisplayConditionName	String	Not null	DisplayCondition elementa nosaukums
Formula	String		Nosacījuma izsteiksme
Loop	Int	FK uz Loop, Not Null	Loop identifikators, kuram pieder nosacījums
DefaultValue	Boolean		Pārbaudes vērība, kas nosaka vai tiek izmantota noklusētā vērtība, ja Formula lauks ir bez vērtības.
IsUseZeroForEmptyParameters	Boolean		Pārbaudes vērtība, kas neeksistējošu atribūtu vietā izmanto vērtību 0

IsChanged			Pārbaudes vērtība, kas nosaka, vai ir bijušas izmaiņas modelī

Tabula "Page"

Tabula satur informāciju par eSales modeļa Page elementiem. Page elements reprezentē tieši vienu tīmekļa lapu.

2.3. tabula

"*Page*" datubāzes tabulas projektējums

Nosaukums	Tips	Papildus Informācija	Apraksts
Page	Int	PK, Not Null, Inc	Page elementa identifikators
Loop	Int	FK uz Loop	Loop identifikators, pie kura pieder Page
Question	Int	FK uz Question	Question identifikators, kas norāda elementa piederību Page
PageName	String		Page elementa nosaukums
PageText	String		Apraksts
PageNumber	Int		Secība datu plūsmas kokā
IsRestricted	Boolean		Pārbaudes vērtība, kas nosaka, vai satura apskatīšanai jābūt autentificētam lietotājam
Template	String		Izklājuma veidnes nosaukums

Layout	String		Izklājuma struktūras nosaukums
RouteName	String		Nosaukums, ar kuru aizvieto URL ceļu
IsChanged	Boolean		Pārbaudes vērtība, kas nosaka, vai ir bijušas izmaiņas modelī

Tabula "Question"

Tabula satur informāciju par eSales modeļa Question elementu. Viens Question elements reprezentē vienu satura vienību noteiktā tīmekļa lapā.

2.4. tabula

"*Question*" datubāzes tabulas projektējums

Nosaukums	Tips	Papildus Informācija	Apraksts
Id	Int	PK, Not Null, Inc	Question elementa idnetifikators
Loop	Int	FK uz Loop, Not Null	Piederība kādam Loop elementam
Page	Int	FK uz Page	Piederība kādam page elementam
Ref	Int	FK uz Question	Question identifikators, kurs ir tiešais vecāks struktūrā konkrētam elementam
Name	String		Nosaukums
Type	String		Kāda kontroles tipa veids, kas reprezentē elementa konstrukciju
Seq	Int		Secība datu plūsmas kokā

Tabula "Attr"

Tabula satur informāciju par eSales modeļa Attr elementu, kur katrs Attr elements apraksta kādu Question elementa satura vai iestatījuma vienību

2.5. tabula

"*Attr*" datubāzes tabulas projektējums

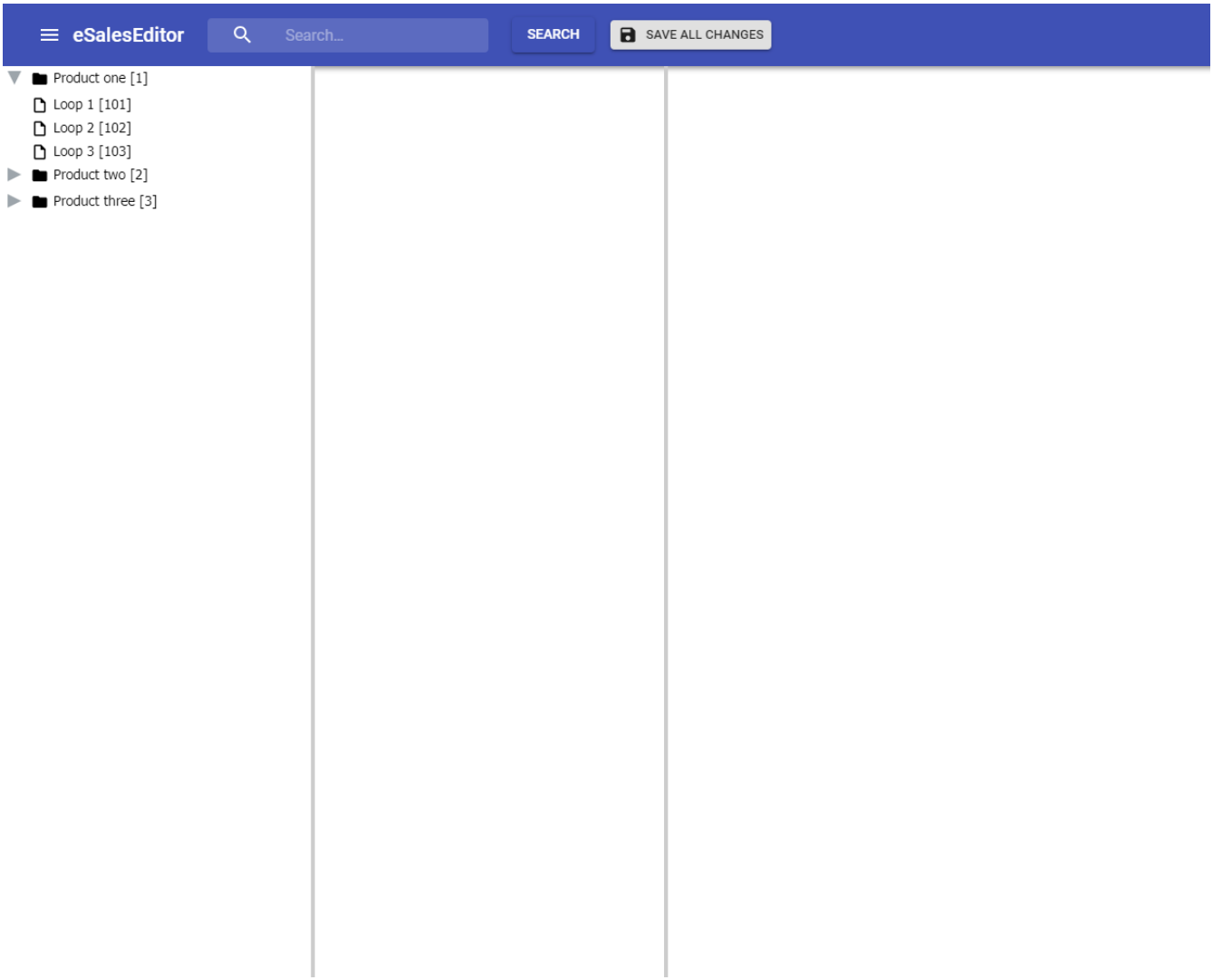
Nosaukums	Tips	Papildus Informācija	Apraksts
Id	Int	PK, Not Null, Inc	DisplayCondition elementa idnetifikators
Ref	Int	FK uz Question,Not Null	Question identifikators, kuram pieder Attr elements
Name	String		Nosaukums
PropertyValue	String		Attr elementa vērtība

2.4 Sistēmas skatu projektējums

Šajā nodaļā ir aprakstīti skatu projektējumi, kas izriet no prasībām par eSales redaktoru.

2.4.1 eSales redaktora sākuma skats

Redaktora sākuma skatu paskaidro lietotājstāsti ar numuriem - #97735



2.10. att. ***eSales redaktora sākuma skats***

Šajā skatā ir attēlots galvenais eSales redaktora skats. Skats ir sadalīts četrās galvenajās daļās – Izvēlnes josla, produktu koka logs, produktu lapu koka logs un redaktora rediģēšanas logs. Kā redzams attēlā (skatīt 2.10. attēlu) zem izvēlnes joslas ekrāns ir sadalīts 3 daļās, kur katra daļa atbilst vienam logam, pirmā attēlo produktu koka logu, otrā produktu lapu koku un trešā redaktora rediģēšanas logu.

2.4.1.1 Izvēlnes josla

Izvēlnes joslu paskaidro lietotājstāsti ar numuriem - #97735, #100207

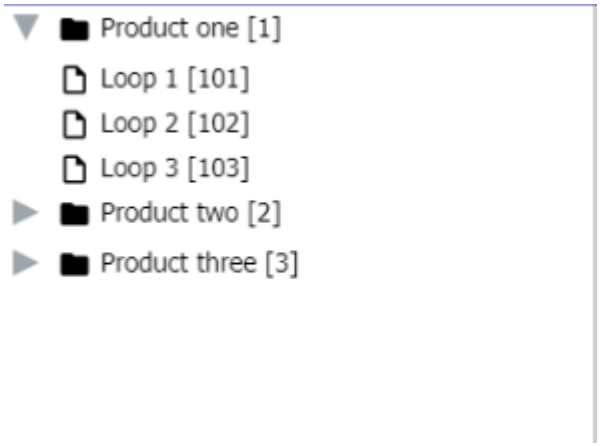
2.11. att. ***eSales redaktora Izvēlnes josla***

Izvēlnes josla vienmēr statiska un nodrošinās vienu un to pašu funkcionalitāti neatkarīgi no ielādētajiem datiem. Uz izvēlnes joslas, sākot ar kreiso pusi, ir attēlots:

- Izvēlnes poga, kas nodrošinātu papildus iestatījumu parādīšanu.
- Redaktora nosaukums.
- Meklēšanas teksta ievades forma.
- Meklēšanas pieprasījuma veikšanas poga.
- Izmaiņu saglabāšanas pieprasījuma veikšanas poga.

2.4.1.2 Produktu koka loga skats

Produkta koka skatu paskaidro lietotājstāsti ar numuriem - #97735, #99434



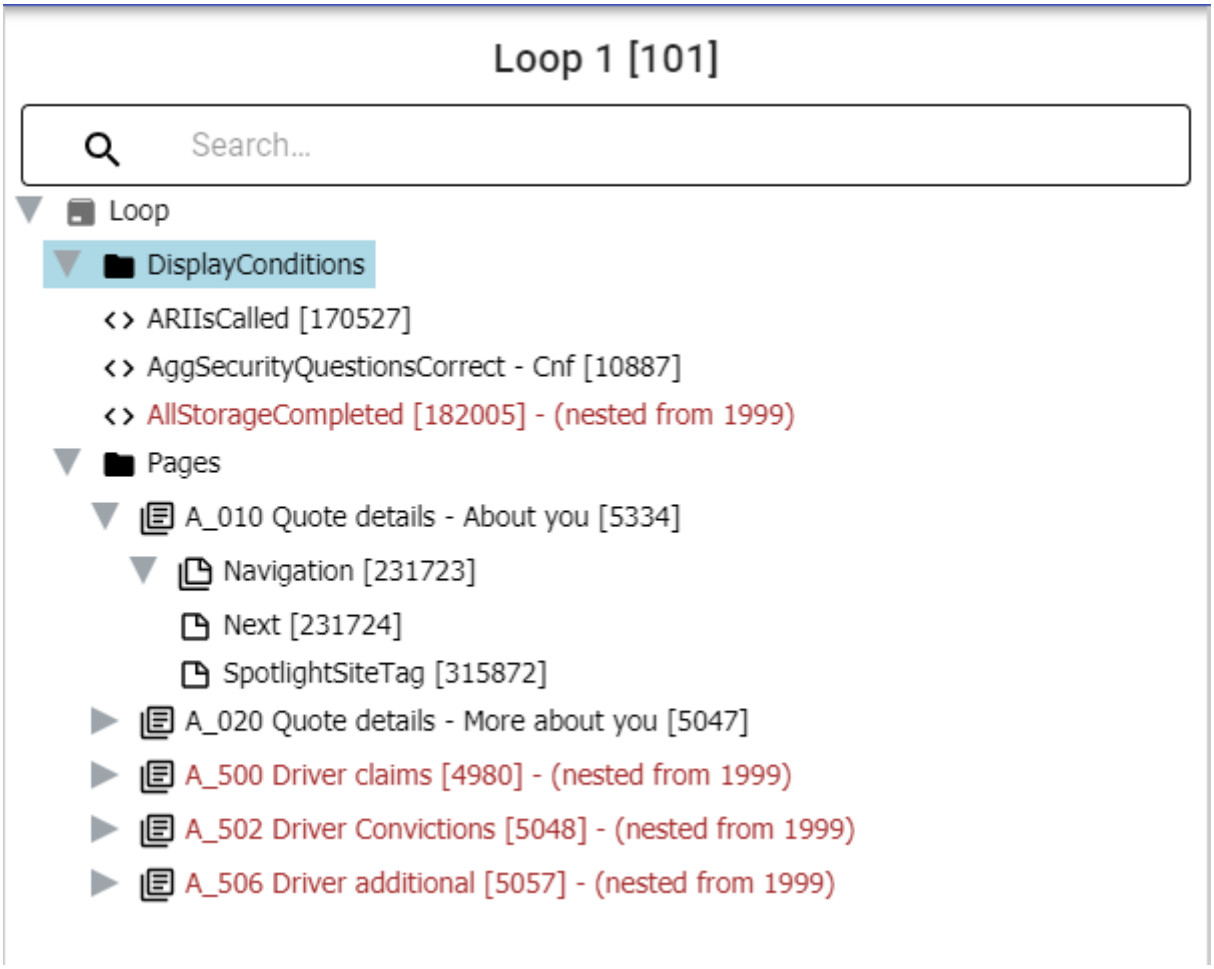
2.12. att. ***eSales redaktora izcelts produktu koka logs***

Produktu koka logs parāda visus ielādētos produktu un ar to saistītos Loop sarakstus. Nospiežot virsū uz kāda no Loop elementiem, tiek atvērts produkta lapu koka logs. Produktu koka loga skatā ir attēlots:

- Koka struktūras produktu elementi ar nosaukumu un identifikatoru
- Koka struktūras Loop elementi ar nosaukumu un identifikatoru
- Atvērts koka struktūras produktu elements

2.4.1.3 Produktu koku lapu loga skats

Produkta lapu koka skatu paskaidro lietotājstāsti ar numuriem - #97735, #97739, #97746, #99434



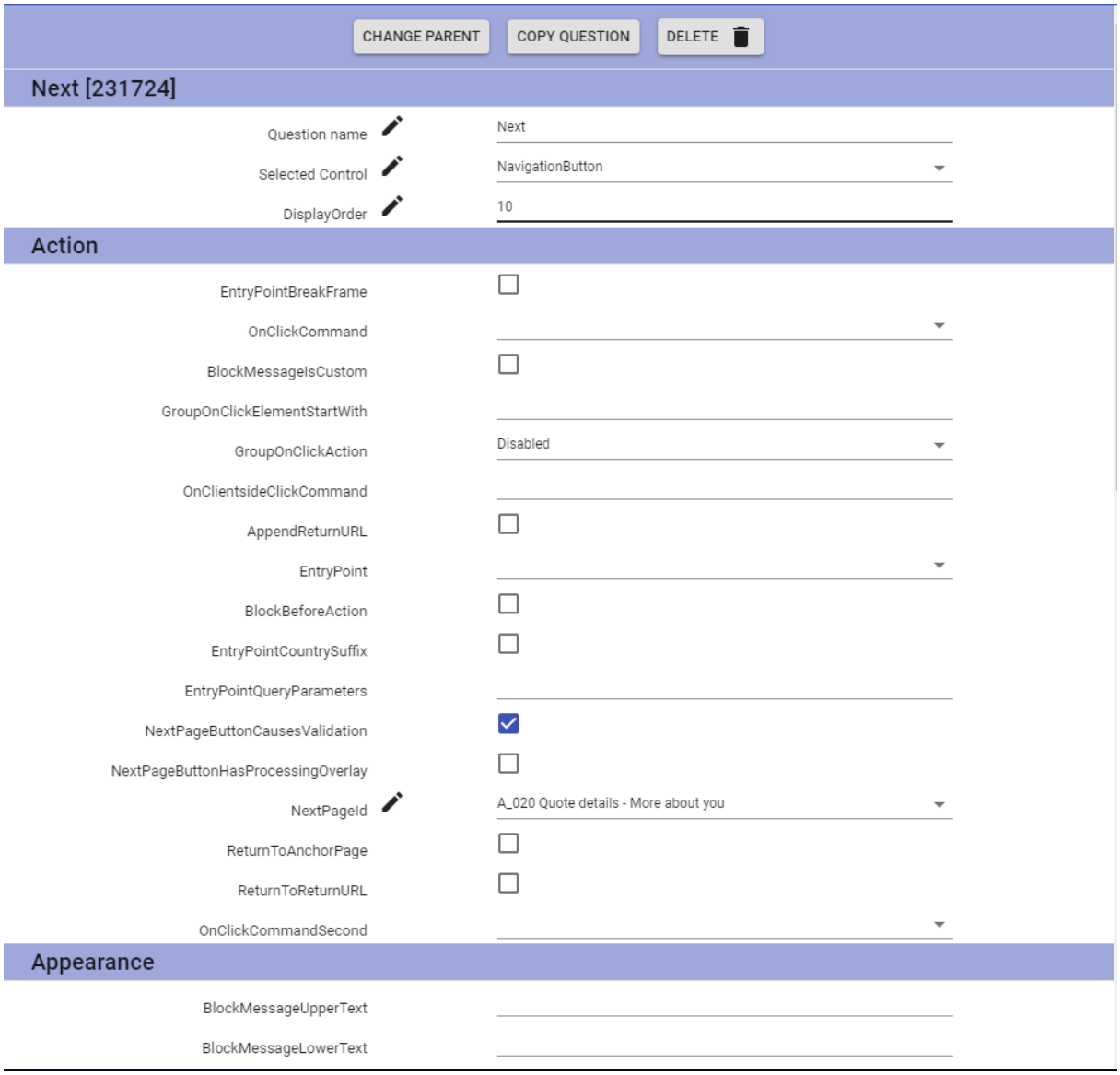
2.13. att. ***eSales redaktora izcelts produktu lapu koka logs***

Produktu lapu koka logs parāda visus konkrēta Loop elementa elementu sarakstu. Katrs no šiem elementiem ir apskatāms sīkāk rediģēšanas logā, kur arī ir iespēja labot to iekšējos iestatījumus. Produktu lapu koka skatā ir attēlots:

- Izvēlētā Loop nosaukums.
- Meklēšanas teksta ievades forma.
- Koka struktūra Loop elementam.
- DisplayCondition, Page un Question elementu nosaukumi ar identifikatoru.
- Ar brūni iekrāsoto krāsu attēloti elementi, kas ir ielādēti papildus no kāda cita Loop elementa, papildus norādot tā identifikatoru.
- Katra elementa tipam ir sava ikona

2.4.1.4 Redaktora Question elementu rediģēšanas loga skats

Redaktora Question elementu rediģēšanas skatu paskaidro lietotājstāsti ar numuriem – #97735, #99252



2.14. att. ***eSales redaktora izcelts Question rediģēšanas logs***

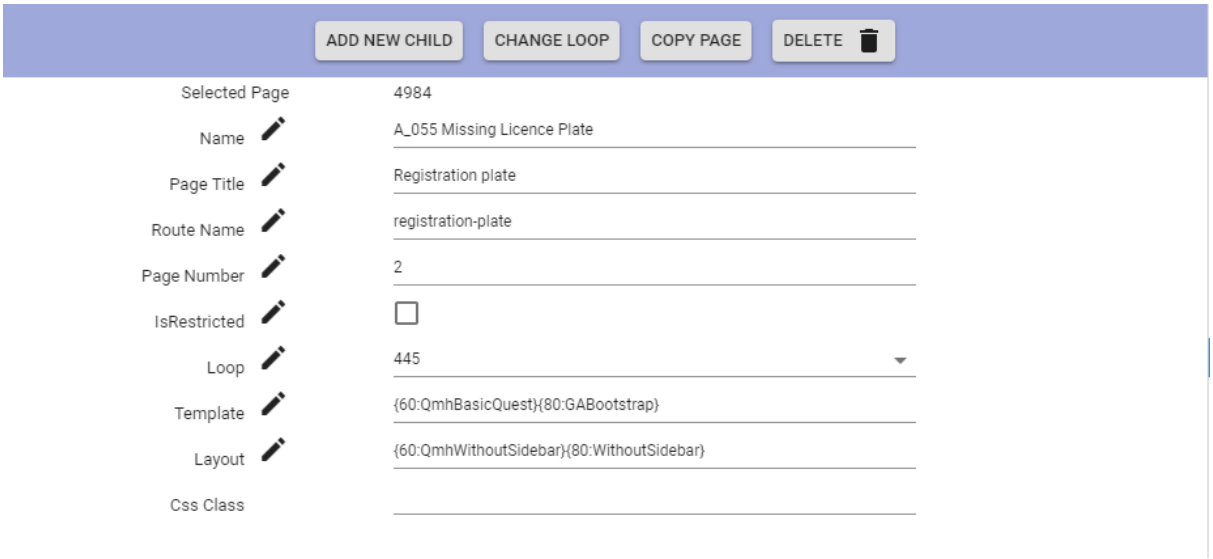
Rediģēšanas logs parāda visus atribūtus kādam elementam ļaujot tos viegli apskatīt un labot. Katrs no atribūtiem tiek iedalīts grupās, lai tos būtu vieglāk atrast, zinot, ka viņi pieder kādai grupai. Katra Question elementa rediģēšanas loga atribūtu saturs ir cits balstoties pēc tā tipa. Rediģēšanas loga skatā ir attēlots:

- Pogu rinda, kur katra poga nodrošina citu funkcionalitāti manipulēšanai ar datiem.
- Galvenes rinda, kur ir attēlots izvēlētā elementa nosaukums un identifikators.
- Zem galvenes rindas ir galvenes iestatījumi, kas ļauj mainīt nosaukumu, Question gadījumā tipu un attēlošanas secību.

- Atribūti sadalīti pa grupām.
- Teksta loga, krītošās izvēlnes un izvēles rūtiņu formas
- Pēc katra atribūta nosaukuma tiek attēlots zīmulītis, ja tā vērtība nav vienāda ar noklusēto vērtību.

2.4.1.5 Redaktora Page, Loop, DisplayCondition rediģēšanas logs

Produkta lapu koka skatu paskaidro lietotājstāsti ar numuriem - #97735, #99252

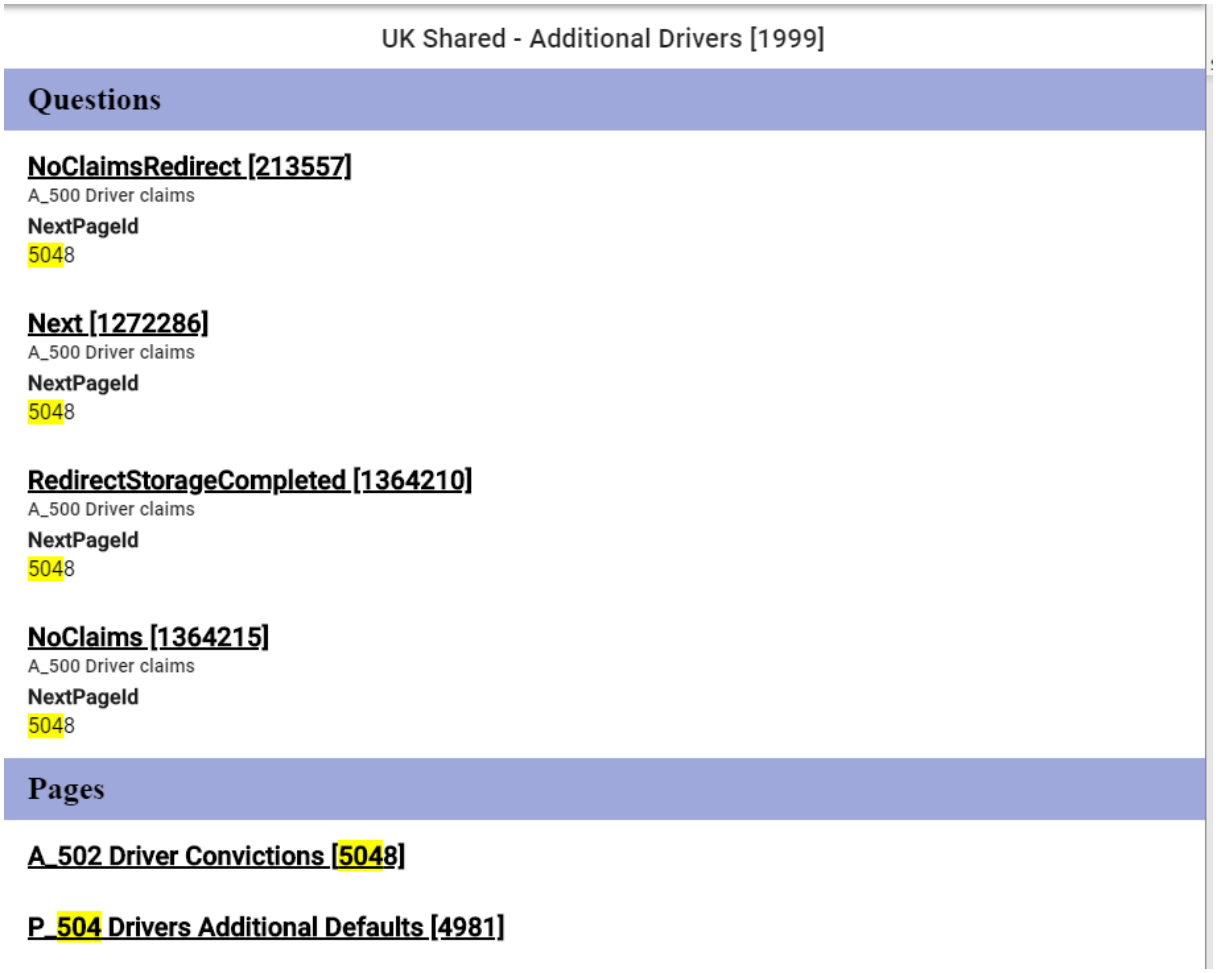


2.15. att. ***eSales redaktora izcelts Page rediģēšanas logs***

Redaktora Page, Loop, DisplayCondition atribūtu saraksts mainās tikai pašu elementu starpā, bet visiem viena tipa elementiem forma ir vienāda. Šo tipu struktūra ir cieti iekodēta, tādēļ atribūti vienmēr ir vieni un tie paši. Šo elementu atribūti netiek grupēti grupās. Šī loga attēlošanas princips ir tāds pats, kā iepriekšējā logā (skatīt sadaļu 2.4.1.4).

2.4.1.6 Redaktora meklēšanas rezultātu loga skats

Produkta lapu koka skatu paskaidro lietotājstāsti ar numuriem - #97735, #100207



2.16. att. ***eSales redaktora izcelts meklēšanas rezultātu logs***

Redaktora meklēšanas rezultātu loga skatā ir attēloti visi meklēšanas rezultāti, kas satur noteiktu frāzi. Nospiežot uz kāda no elementu nosaukumiem, tiek automātiski atvērts rediģēšanas logs ar izvēlēto elementu un tā atrašanās vieta kokā. Meklēšanas rezultātu skatā ir attēlots:

- Meklētā Loop elementa nosaukums un ID.
- Meklēšanas rezultātu grupēšana pēc elementa tipa.
- Izcelts un pasvītrots elementa nosaukums kopā ar ID.
- Question elementiem papildus zem nosaukuma tiek norādīts Page elements, uz kuru tas attiecas.
- Atribūta nosaukums, zem kura ir atribūta vērtība.
- Dzelteni iekrāsotas atrastās vietas, kas satur meklēšanas frāzi.

3. TESTĒŠANAS DOKUMENTĀCIJA

3.1 Ievads

Šajā nodaļā tiek aprakstīts izstrādāto sistēmas daļu testēšana un testēšanas rezultāti. Testēšanai tika izmantoti vienībtesti ņemot vērā programmatūras iekšējo struktūru un ārējo saskarnes darbību. Testi tika izpildīti regulāri veicot jebkādas izmaiņas.

3.2 Testējamās raksturiezīmes

Tiek pārbaudīts vai izstrādāto klašu metodes veic korektu darbību atbilstoši prasībām. Testi ir veidoti katram elementam sistēmā un tiek testēta gan datu apstrādes funkcionalitāte, gan saskarnes ģenerēšana

3.3 Testpiemēru specifikācija

Izstrādes procesā izmantotie testi tiek attēloti tabulās ar šādu informāciju:

- Testa nosaukums
- Ievaddati
- Sagaidāmais rezultāts
- Rezultāts – ja rezultāts sakrīt ar gaidīto, tas tiek atzīmēts ar “+”, ja tests kādā no testēšanas reizēm izgāzās, tiks aprakstīts izgāšanās iemesls.
- Saistītā lietotājstāsta numurs

Tā kā pie katras izmaiņas automātiski tika izpildīti attiecīgie testi, vieglākai testu pārskatāmībai, tie tiks uzrādīti vienreiz. Metodes, kuru testu rezultāts neatgrieza sagaidīto rezultātu, tika uzreiz izlabotas.

3.4 Testēšanas rezultāti

3.4.1 Datu modeļu klašu testi

3.1. tabula

Datu modeļu klašu testu rezultāti

Tests	Ievaddati	Sagaidāmais rezultāts	Rezultāts	#Numurs
Ielādēt datus no neeksistējošas direktorijas.	Norāde uz neeksistējošu direktoriju.	Tiek dots kļūdas paziņojums.	+	#97731
Datu ielāde no XML faila.	XML testa fails.	Atgriezts objekts ar pareizi sadalītiem datiem.	Pirmajās reizēs kļūdaini ielasīti dati.	#97731
Datu saglabāšana XML datnēs.	Testa datu kopa.	Atgriezts identisks fails orģinālam.	+	#97731
Visu Page elementu saraksta atgriešana konkrēta Loop ietvaros.	Loop elements.	Sakārtots saraksts ar Page elementiem.	Neeksistējošu Page elementu gadījumā, netika pareizi turpināta programmas darbība.	#97744
Visu DisplayCondition elementu saraksta atgriešana konkrēta Loop ietvaros.	Loop elements.	Sakārtots DisplayCondition saraksts.	+	#97731
Visu Question elementu atgriešana konkrēta Page ietvaros.	Page elements.	Pēc identifikatora sakārtots Question saraksts.	+	#97744
Jaunu Question, Attr, Page, DisplayCondition elementu izveide.	Testa datu kopa.	Jauni pievienotu elementu ID ir negatīvs.	+	#97742

Negatīvu ID atjaunošana pret īstiem ID.	Testa datu kopa.	Elementu negatīvie ID aizstāti ar jauniem unikāliem ID. Gadījumā ja negatīvu elementu ir vairāk, nekā pieprasīts no DB tiek brīdināts lietotājs par kļūdu.	Gadījumā, kad negatīvu elementu ir vairāk nekā jauni pieprasīto ID, netiek brīdināts lietotājs un tiek turpināta darbība ar nepareiziem ID.	#97732
Pārbauda vai Question elementam eksistē noteikts Attr elements	Question elements, Attr elementa vārds	Tiek atgriezta elementa vērtība	+	#97732
Pieprasa Question elementam neeksistējošu Attr vērtību.	Question elements, Attr elementa vārds.	Tiek atgriezta tukša vērtība.	Tukšas vērtības vietā tiek atgriezts Null.	#97732

3.4.2 Meklēšanas testi

3.2. tabula

Meklēšanas testi

Tests	Ievaddati	Sagaidāmais rezultāts	Rezultāts	#Numurs
Tukša meklēšanas frāze.	Testu datu kopaTukša meklēšanas frāze.	Paziņojums par frāzes ievadīšanu.	Atgriezts pilnīgi visa datu kopa.	#100207
Meklēšana pēc konkrētas frāzes.	Testu datu kopa, Meklēšanas frāze.	Tiek atgrieztas iekrāsotas, pa grupām sadalītas, elementu vērtības, kuras satur meklēšanas frāzi.	+	#100207
Meklēšana pēc konkrētas frāzes visos Loop elementos.	Testu datu kopa, meklēšanas frāze, visu Loop meklēšanas iestatījums.	Tiek atgrieztas visas elementu vērtības, sagrupētas pēc Loop elementiem.	+	#100207
Meklēšana tikai pēc DisplayCondition elementa tipa.	Testu datu kopa, meklēšanas frāze, DisplayCondition elementa tips.	Tiek atgriezti visi konkrētā tipa rezultāti, kas satur meklēšanas frāzi.	+	100207

Meklēšana tikai pēc Page elementa tipa.	Testu datu kopa, meklēšanas frāze, Page elementa tips.	Tiek atgriezti visi konkrētā tipa rezultāti, kas satur meklēšanas frāzi.	+	#100207
Meklēšana tikai pēc Question elementa tipa.	Testu datu kopa, meklēšanas frāze, Question elementa tips.	Tiek atgriezti visi konkrētā tipa rezultāti, kas satur meklēšanas frāzi.	+	#100207
Meklēšana pēc regulārās izteiksmes.	Testu datu kopa, regulārā izteiksme.	Tiek atgriezti visi rezultāti, kas satur regulāro izteiksmi.	+	#100207

3.4.3 Komponenšu attēlošanas testi

3.3. tabula

Komponenšu attēlošanas testu rezultāti

Tests	Ievaddati	Sagaidāmais rezultāts	Rezultāts	#Number
Produktu koka ģenerēšana.	Testa datu kopa.	Atgriezts no padotās datu kopas izveidots pareizi strukturēts produktu koks ar pareizu saturu.	+	#97735
Produkta lapu koka ģenerēšana.	Testa datu kopa, Loop identifikators.	Atgriezts no padotās datu kopas produkta lapu koks, ar pareizu struktūru un atbilstošu saturu.	+	#97735
Koka ģenerēšana ar tukšu datu kopu.	Testa datu kopa.	Atgriezts kļūdas paziņojums par neeksistējošu datu kopu.	Netiek atgriezts kļūdas paziņojums un tiek attēlota tukša lapa.	#97735
Question elementu formas ģenerēšana.	Testa datu kopa, Question identifikators.	Atgriezta Question forma, ar pareizu atribūtu sarakstu.	Nesaturēja visus vajadzīgos atribūtus.	#99252
DisplayCondition elementu formas ģenerēšana.	Testa datu kopa, DisplayCondition identifikators.	Atgriezta DisplayCondition forma ar pareizu atribūtu sarakstu.	+	#99252
Page elementu formas ģenerēšāna.	Testa datu kopa, Page identifikators.	Atgriezta Page forma ar pareizu atribūtu sarakstu.	+	#99252

Loop elementa formas ģenerēšana.	Testa datu kopa, Loop indetifikators.	Atgriezta Loop forma ar pareizu atribūtu sarakstu.	+	#99252
Jaunas Question formas ģenerēšana pie tipa maņas.	Testa datu kopa, Question identifikators, tips.	Atgriezta Question forma ar jaunu atribūtu sarakstu.	+	#97745
Jauna Question elementa pievienošana.	Testa datu kopa, Question identifikators.	Atgriezta jaunā Question forma un produktu lapu koka struktūras atjaunošana ar jaunā elementa iezīmēšanu.	Netiek pareizi atjaunots produktu lapu koks.	#97742
Question elementa kopēšana.	Testa datu kopa, Question identifikators.	Atgriezts kopētā Question forma, atjaunota produktu lapu koka struktūra ar visiem kopētā elementa bērniem.	+	#97742
Question elementa dzēšana.	Testu datu kopa, Question identifikators.	Elements tiek dzēsts no datu modeļa, produktu lapu koka struktūra tiek atjaunota.	Dzēsta elementa vērtība netika pilnībā dzēsta atstājot tukšu elementu masīvā.	#97747
Question elementa vecāka maiņa.	Testu datu kopa, Question identifikators.	Izvēlētais Question elements tiek dzēsts no vecā vecāka un pievienots jaunajam vecākam. Pareiza produktu lapu koka atjaunota struktūra.	Elements netiek pareizi pārvietots koka struktūra.	#97744
Noklusēto datu slēpšana rediģēšanas formās.	Testu datu kopa, slēpt noklusēto vērtību iestatījums.	Atgriezti atribūti, kuru vērtības nav vienādas ar noklusētajām vērtībām.	+	#99252
Teksta formu ievadīto datu saglabāšana.	Testu datu kopa.	Jauno vērtību saglabāšana datu modelī. Datu pareiza attēlošana formās.	Formu, kuru vērtības nebija tukšas, vērtības nemainījās.	#97732
Izvēlnes rūtiņu formu stāvokļa maiņā	Testa datu kopa.	Jauno vērtību saglabāšana datu modelī. Pareiza formas atjaunošana pēc stāvokļa maiņas.	+	#97732

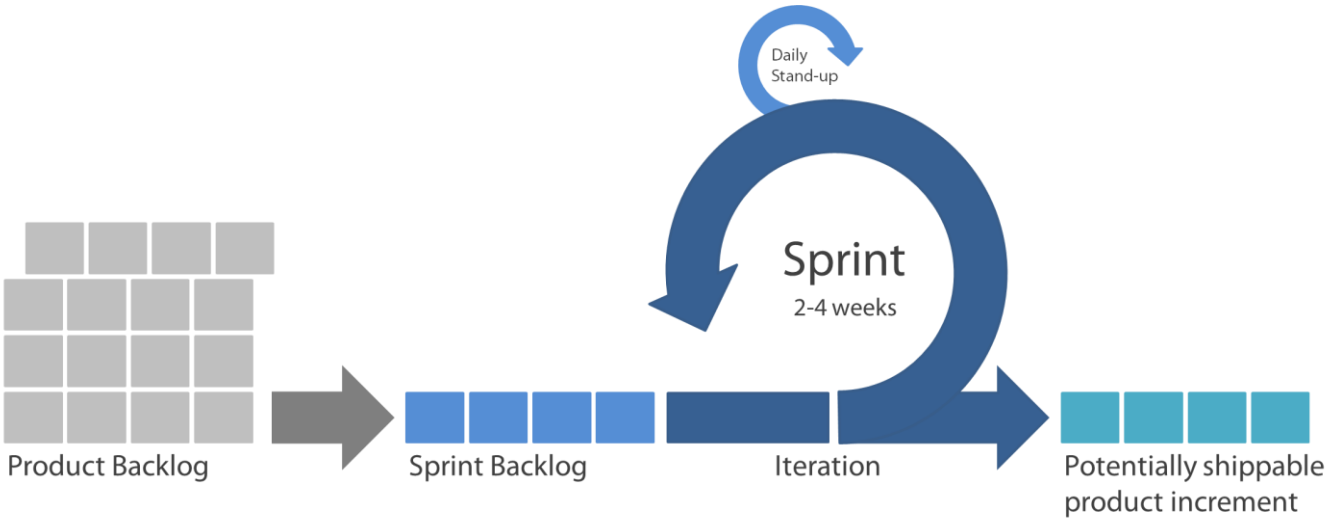
Krītošas izvēlnes formu pareiza attēlošana.	Testa datu kopa.	Atgriezta krītošās izvēlnes forma ar pareiziem izvēlnes datiem.	Tiek atgriezta forma bez vērtībām.	#99252
Krītošas izvēlnes formu datu maiņas saglabāšana	Testa datu kopa, izvēlētās opcijas identifikators	Jaunās vērtības saglabāšana datu modelī. Pareiza formas atjaunošana	+	#97732

4. PROJEKTA ORGANIZĀCIJA

Darba izstrāde tika veikta jau esoša projekta ietvaros, kur izstrāde notiek pēc spējās programmatūras izstrādes metodoloģijas, kura ir piemērota, ja darba izstrādes sākumā nav zināmas visas, sistēmā izstrādājamās, prasības. Projektu izstrādāja viens cilvēks.

Spējā izstrādē tika izmantots SCRUM modelis, kur izstrāde tikai veikta iterācijās un vienas iterācijas ilgums ir divas nedēļas, kuru sauc par sprintu. Katra sprinta sākumā tika izrunāti sprintā paveicamie darbi un ko vajadzētu prioritizēt izstrādes procesā. Visas prasības tiek aprakstītas lietotājstāstu formā. Projekta iekšienē izstrādātāji tika sadalīti pa komandām un katrai komandai ir vadītājs. Katru dienu tika veidotas komandu sapulces, kur katrs komandas loceklis dalījās ar to, jo ir paveicis iepriekšējā dienā un ar ko strādās tālāk. Visas komandu sapulces notika attālināti. Katras iterācijas beigās tika veidota papildus sapulce, kurā katrs komandas loceklis pauda savu viedokli par aizvadīto iterāciju un ieteikumus nākamajām iterācijām.

Visa projekta vadībai un organizācijai tika izmantots Redmine, kur ir iespējams veidot lietotājstāstus, seko izstrādes progresam, patērētajam laikam un konkrētām koda izmaiņām saistībā ar kādu lietotājstāstu. Katram lietotajstāstam komanda iedeva izstrādes procesam paredzētu laiku un svarīguma pakāpi.



4.1. att. ***SCRUM izstrādes procesa diagramma***[6]

5. KVALITĀTES NODROŠINĀŠANA

Izstrādātās produkta kvalitātes nodrošināšanai, tika veidota programmatūras dokumentācija, kurā tika dokumentētas programmatūras prasības, programmatūras projektējums un testēšanas dokumentācija. Veidojot dokumentāciju tika ņemti par piemēru dažādas uzņēmuma vadlīniju paskaidrojoši dokumenti.

Papildus kvalitātei īpaša uzmanība tika pievērsta testēšanai. Testēšana tika veikta rakstot vienībtestus, kas tika automātiski izpildīti pie izmaiņu veikšanas kodā. Veicot jaunu komponenšu izstrādi tika veikta manuāla komponenšu testēšana, kas ātri ļauj identificēt procesā izveidotās kļūdas. Papildus programmatūra tiek dota testēt citiem programmētājiem.

Visas izstrādātāja izmaiņas kodā tiek apskatītas, kur apskates veic kāds izstrādātājs no tās pašas komandas. Ja pēc apskates ir identificēti uzlabojumi vai atrastas kļūdas, lietotājstāsts tiek atgriezts sākotnējam izstrādātājam kļūdu labošanai vai uzlabojumu veikšanai.

Tiek ievērota vispārpieņemtā programmēšanas labā prakse attiecīgajai programmēšanas valodai. Papildus tiek lietoti spraudņi kā *TSLint* un *Prettier,* kas reālā laikā analizē kodu parādot kļūdas, vai pie saglabāšanas formatē kodu vieglākai pārskatīšanai.

Papildus tiek izmantota programmatūras versiju pārvaldība *Git*, kur tiek glabātas visas izmaiņas un nepieciešamības gadījumā var atjaunot kādu no vecākām, strādājošām versijām.

6. KONFIGURĀCIJU PĀRVALDĪBA

Produkta izstrādes laikā projektu konfigurācijas pārvaldībai un versiju kontrolei tika lietots Git.

Katra izstrādes vienuma beigās tiek izmantots Git, lai saglabātu izmaiņas. Versiju kontrole tika veikta pēc *Git Flow Branching* modeļa. Koda, konfigurācijas un citu datņu izmaiņas tiek sūtītas uz glabātas uz kopēja servera, kur nepieciešamības gadījumā ir iespējams salīdzināt divas versijas un redzēt izmaiņās, kā arī atjaunot vienu no vecākajām versijām. Šim nolūkam tiek izmantots *GitLab*. Katrai izmaiņai tiek piešķirts revīzijas numurs. Revīzijas tiek piesaistītas Redmine projektu pārvaldības sistēmas lietotājstāstiem, kas padara revīziju vēsturi labi pārskatāmu un saprotamu. Projekta iekšienē tiek ievērots princips, ka izmaiņas drīkst ievietot tikai lietotājstāsta ietvaros, lai būtu saprotams izmaiņu nolūks.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

Kvalifikācijas darba autors darba sākumā novērtēja lietotājstāstus ar aptuvenu, uzdevumu izpildei vajadzīgo laiku, konsultējoties ar darba vadītāju. Veicot novērtējumu autors papildus vadījās pēc savām zināšanām un fakta, ka izstrādē tiks izmantotas jaunas tehnoloģijas.

7.1 tabula

Darbietilpības novērtējuma tabula

Lietotājstāsta numurs	Prognozētais laiks (stundās)	Patērētais laiks (stundās)
#97731	40	49
#97732	40	42.25
#97734	16	31.5
#97735	104	138.75
#99252	48	63.25
#97739	8	12.75
#97742	24	31
#97744	32	33
#97745	4	3.5
#97746	8	6
#97747	4	4.75
#99434	4	3.5
#100207	32	25.5
Kopā:	**366**	**443.75**

Rezultātā, pirms izstrādes uzsākšanas darbietilpības novērtējums tika novērtēts aptuveni uz 366 stundām. Pēc izstrādes tika secināts, ka darbietilpība ir pārsniegusi prognozēto laiku, kā rezultātā bija nepieciešamas 443.75 stundas. Kopumā izstrādātais projekts ir vērtējams 3 personmēnešu darbietilpībai.

SECINĀJUMI

Kvalifikācijas darba beigu rezultāts ir eSales redaktora uzlabojums, kas uzlaboja redaktora ātrdarbību. Redaktors ielādes laiku uzlaboja līdz pat 10 sekundēm. Vecajā redaktorā, produkta elementa ielāde prasa vidēji 5 sekundes, kur jaunā redaktora ielādes laiks ir vidēji ir 0.05 sekundes. Kas ir vismaz par 5 sekundēm ātrāk. Tika uzlabota arī lietojamība, atvieglojot apdrošināšanas sistēmas izstrādātāju darbu. Papildus tika palielināta rīka uzturamība, veidojot to uz jaunām tehnoloģijām un ieviešot lielu modularitāti, kas ļauj nākotnē viegli pievienot papildus funkcionalitāti. Izstrādātajai programmatūrai tika izveidota atbilstoša dokumentācija, kurā tiek aprakstītas izstrādājamās sistēmas prasības, projektējums, testēšana un rezultāti.

Izstrādājot darbu pēc spējās izstrādes metodoloģijas, tika secināts, ka izstrādātājam ir jābūt spējīgam pielāgoties pie dažādām prasību izmaiņām, kuras var rasties izstrādes procesa laikā. Svarīga arī ir komunikācija ar klientu, kas palīdz novērst pārpratumus un palielina sapratni par gaidāmo rezultātu.

Kvalifikācijas darba autoram, strādājot pie darba izstrādes, tika gūtas zināšanas un pieredze darbā ar:

- Electron izstrādes rīku
- Node.js
- React satvaru
- TypeScript programmēšanas valodu
- Material UI dizainu
- XML datņu lietošanu

Projekta ietvaros tika stiprinātas komunikācijas prasmes, kā arī gūta pieredze darbā ar komandu. Papildus tika uzlabotas teorētiskās zināšanas programmēšanā.

IZMANTOTĀ LITERATŪRA

1. Akadēmiskā terminu datu bāze [tiešsaiste]. – [atsauce 18.05.2019]. Pieejams: http://termini.lza.lv/
2. Git dokumentācija [tiešsaiste]. – [atsauce 18.05.2019]. Pieejams: https://git-scm.com/docs
3. Electron dokumentācija [tiešsaiste]. – [atsauce 18.05.2019]. Pieejams: https://electronjs.org/docs
4. React dokumentācija [tiešsaiste]. – [atsauce 18.05.2019]. Pieejams: https://reactjs.org/docs/getting-started.html
5. Node.js dokumentācija [tiešsaiste]. – [atsauce 18.05.2019]. Pieejams: https://nodejs.org/en/docs/
6. Redmine dokumentācija [tiešsaiste]. [atsauce 18.05.2019]. Pieejams: http://www.redmine.org/projects/redmine/wiki

PIELIKUMI

1. pielikums
Loop modeļa klase

```
import { Page } from './Page';
import { Question } from './Question';
import { Attr } from './Attr';
import { DisplayCondition } from './DisplayCondition';

class Loop {
    public loop: string; // number
    public productIds: string;
    public loopName: string;
    public routeName: string;
    public defaultPageId: string; // number
    public loadEntryPageId: string;
    public languageId: string;
    public metaTags: string;
    public useValidationSummary: string;
    public useLoopCss: string;
    public template: string;
    public layout: string;
    public nestedLoopId = '';
    public folderName: string;
    public pages: Page[] = [];
    public displayCondition: DisplayCondition[] = [];
    public isChanged = false;
    public fileCount = 0;
    public filesLoaded = 0;
    public isMapped = false;
    public productValues: object[];
    [key: string]: any;

    public mapQuestions() {
        const HashMap = require('hashmap');
        this.pages.forEach(page => {
            const siblings = new HashMap();
            const root = new HashMap();
            const notRoot = new HashMap();
            const forDeletion: any[] = [];

            page.questions.forEach(q => {
                const node = q;
                node.question = siblings.get(node.id);

                if (node.ref === null) {
```

```
                    root.set(node.id, node);
                } else {
                    notRoot.set(node.id, node);
                    let posSib = siblings.get(node.ref);
                    forDeletion.push(node.ref);
                    if (posSib) {
                        posSib.push(node);
                    } else {
                        posSib = [node];
                        siblings.set(node.ref, posSib);
                        const posPar = notRoot.get(node.ref) || root.get(node.ref);
                        if (posPar) {
                            posPar.question = posSib;
                        }
                    }
                }
            });
            page.questions = page.questions.filter(item =>
!forDeletion.includes(item.ref));
        });
        this.isMapped = true;
        this.pages.sort(this.compare);
    }

    public compare(a: Page, b: Page) {
        if (a.pageName < b.pageName) {
            return -1;
        }
        if (a.pageName > b.pageName) {
            return 1;
        }
        return 0;
    }

    public getPageList() {
        const pageList: object[] = [];
        this.pages.forEach(page => {
            pageList.push({
                id: page.id,
                name: page.pageName,
                fullName: `${page.pageName} [${page.id}]`,
            });
        });
        return pageList;
    }

    public getDisplayConditionList() {
        const conditionList: any[] = [];
        this.displayCondition.forEach(condition => {
```

```
            conditionList.push(condition);
        });
        return conditionList;
    }

    public getDisplayConditionFormulaValue(id: string) {
        const attr = this.displayCondition.find(condition => condition.id === id);
        return attr ? attr.formula : null;
    }

    public getAllLoopQuestions(exception: Question = null) {
        let questions = [] as Question[];
        this.pages.forEach(page => {
            questions = questions.concat(page.getAllQuestionArray());
        });
        exception ? (questions = questions.filter(item => exception.id !== item.id)) :
null;
        return questions;
    }

    public getAllGroupControlQuestions(exception: Question = null) {
        let questionsList = this.getAllLoopQuestions(exception);
        questionsList = questionsList.filter(item => item.type === 'GroupControl');
        return questionsList;
    }

    search(searchParams: any, attributeNames: any) {
        var regex = new RegExp(searchParams.searchValue, 'i');
        var childrenExist = false;
        var v = [
            {
                name: 'DisplayConditions',
                children: [] as any,
            },
            {
                name: 'Questions',
                children: [] as any,
            },
            {
                name: 'Pages',
                children: [] as any,
            },
        ];

        var dcList =
            searchParams.searchGroup == 'All' || searchParams.searchGroup ==
'DisplayConditions'
                ? this.getDisplayConditionList()
                : null;
```

```
        dcList
            ? dcList.forEach((condition: DisplayCondition) => {
                  var DCElement = {
                      id: condition.id,
                      name: condition.fullName,
                      attributes: [] as any,
                      nameIsMatch: false,
                  };
                  if (condition.fullName.search(regex) != -1) {
                      DCElement.nameIsMatch = true;
                  }
                  Object.keys(condition).forEach(key => {
                      if (typeof condition[key] == 'string' && !(key == 'id' || key ==
'name' || key == 'fullName')) {
                          if (condition[key].search(regex) != -1) {
                              DCElement.attributes.push({
                                  name: key,
                                  value: condition[key],
                              });
                          }
                      }
                  });

                  if (DCElement.attributes.length > 0 || DCElement.nameIsMatch) {
                      v[0].children.push(DCElement);
                      childrenExist = true;
                  }
              })
            : null;

        var questionList =
            searchParams.searchGroup == 'All' || searchParams.searchGroup == 'Questions'
                ? this.getAllLoopQuestions()
                : null;
        questionList
            ? questionList.forEach((question: Question) => {
                  var QElement = {
                      id: question.id,
                      name: question.fullName,
                      pageName: question.pageName,
                      attributes: [] as any,
                      nameIsMatch: false,
                  };

                  if (question.fullName.search(regex) != -1) {
                      QElement.nameIsMatch = true;
                      childrenExist = true;
                  }
```

```
                question.attr
                    ? question.attr.forEach((atr: Attr) => {
                          if (atr.value) {
                              if (atr.checkEditorType(question.type) ==
'ProductAttributeSelector') {
                                  var check = attributeNames.find((attr: any) => attr.Id
== atr.value);
                                  if (check) {
                                      QElement.attributes.push({
                                          name: atr.controlProperty,
                                          value: `${check.Name} [${check.Id}]`,
                                      });
                                  }
                              } else if (atr.value.search(regex) != -1) {
                                  QElement.attributes.push({
                                      name: atr.controlProperty,
                                      value: atr.value,
                                  });
                              }
                          }
                      })
                    : null;

                if (QElement.attributes.length > 0 || QElement.nameIsMatch) {
                    v[1].children.push(QElement);
                    childrenExist = true;
                }
            })
        : null;

        var pageList =
            searchParams.searchGroup == 'All' || searchParams.searchGroup == 'Pages' ?
this.getPageList() : null;
        pageList
            ? pageList.forEach((page: any) => {
                  if (page['fullName'].search(regex) != -1) {
                      v[2].children.push({
                          name: page.fullName,
                          id: page.id,
                          attributes: [] as any,
                      });
                      childrenExist = true;
                  }
              })
            : null;

        return childrenExist ? v : null;
    }
}export { Loop };
```

Page modeļa klase

```
import { Question } from './Question';
class Page {
    id: any;
    loopId: any;
    pageName: any;
    questions: Question[] = [];
    isHeaderVisible: any;
    layout: any;
    pageCssClass: any;
    pageNumber: any;
    pageText: any;
    routeName: any;
    template: any;
    isChanged = false;
    [key: string]: any;

    public getAllQuestionArray() {
        let array = [] as Question[];
        this.questions.forEach(question => {
            array = array.concat(question.getQuestions());
        });

        array.sort(this.compare);
        return array;
    }

    public compare(a: Question, b: Question) {
        if (Math.abs(parseInt(a.id, 10)) < Math.abs(parseInt(b.id, 10))) {
            return -1;
        }
        if (Math.abs(parseInt(a.id, 10)) > Math.abs(parseInt(b.id, 10))) {
            return 1;
        }
        return 0;
    }

    public removeQuestionFromParent(question: Question) {
        if (question.ref == null) {
            this.questions.splice(this.questions.findIndex(x => x.id === question.id), 1);
        }
        const arr = this.getAllQuestionArray();
        arr.forEach(q => {
            if (question.ref === q.id) {
                q.question.splice(q.question.findIndex(x => x.id === question.id), 1);
            }
        });
```

```
    }

    public copyQuestionToParent(question: Question) {
        if (question.ref == null) {
            this.questions.push(question);
        }
        const arr = this.getAllQuestionArray();
        arr.forEach(q => {
            if (question.ref === q.id) {
                q.question.push(question);
            }
        });
    }
}

export { Page };
```

3. pielikums

Koku būvēšanas klase

```
import { LoopList } from '../renderer/model/LoopList';
import { ProductAttributeHelper } from './ProductAttributeHelper';
import { Loop } from '../renderer/model/Loop';
import { Question } from '../renderer/model/Question';
import { Page } from '../renderer/model/Page';

export class TreeBuilder {
    public getProductTree(loopList: LoopList) {
        let tree: any[] = [];
        const helper = new ProductAttributeHelper();
        const productTree = helper.getProductNames();
        productTree.forEach((product: any) => {
            const children: object[] = [];
            loopList.loops.forEach(loop => {
                if (loop.productIds.split(';').includes(product['id'])) {
                    children.push({
                        id: loop.loop,
                        name: `${loop.loopName} [${loop.loop}]`,
                        loop: loop.folderName,
                        type: 'loop',
                    });
                }
            });
            tree.push({
                id: product.id,
                name: `${product['Name']}  [${product['id']}]`,
                type: 'product',
                children,
            });
        });
```

```
    tree = tree.filter((item: any) => item.children.length > 1);
    return tree;
}

public buildTree(loop: Loop, nestedLoop: Loop = null) {
    const tree = [
        {
            id: 'Loop',
            name: 'Loop',
            fullName: 'Loop',
            type: 'loop',
            obj: loop,
            toggled: true,
            children: [
                {
                    id: 'DisplayCondition',
                    name: 'DisplayConditions',
                    fullName: 'DisplayConditions',
                    type: 'group',
                    children: [] as object[],
                },
                {
                    id: 'Pages',
                    name: 'Pages',
                    fullName: 'Pages',
                    type: 'group',
                    toggled: true,
                    children: [] as object[],
                },
            ],
        },
    ];

    this.addDisplayConditionNodes(loop, tree);
    this.addPageNodes(loop, tree);

    if (nestedLoop) {
        this.addDisplayConditionNodes(nestedLoop, tree, true);
        this.addPageNodes(nestedLoop, tree, true);
    }

    this.sortTree(tree);

    return tree;
}

public sortTree(tree: any) {
    tree[0].children[0].children.sort(this.compare);
    tree[0].children[1].children.sort(this.compare);
```

```
    }

    public compare(a: any, b: any) {
        if (a.fullName < b.fullName) {
            return -1;
        }
        if (a.fullName > b.fullName) {
            return 1;
        }
        return 0;
    }

    public addDisplayConditionNodes(loop: Loop, tree: any, isNested: boolean = false) {
        loop.displayCondition.forEach(condition => {
            const jsonCondition = {
                ref: 'Loop',
                id: condition.id,
                name: condition.name,
                fullName: `${condition.name} [${condition.id}] ${isNested ? ` - (nested
from ${loop.loop}) ` : ''}`,
                type: 'displayCondition',
                obj: condition,
                isNested,
            };

            tree[0].children[0].children.push(jsonCondition);
        });
    }

    public addPageNodes(loop: Loop, tree: any, isNested: boolean = false) {
        loop.pages.forEach(page => {
            const jsonPages = {
                ref: 'Pages',
                id: page.id,
                name: page.pageName,
                fullName: `${page.pageName} [${page.id}] ${isNested ? ` - (nested from
${loop.loop}) ` : ''}`,
                type: 'page',
                obj: page,
                isNested,
                children: [] as object[],
            };

            page.questions.forEach(question => {
                jsonPages.children.push({
                    ref: page.id,
                    id: question.id,
                    name: question.name,
                    fullName: `${question.name} [${question.id}]`,
```

```
                    type: 'question',
                    page,
                    obj: question,
                    isNested,
                    children: this.qChild(question.question, page, isNested),
                });
            });
            tree[0].children[1].children.push(jsonPages);
        });
    }

    public buildTreeForChangeParent(questionToExclude: Question, loop: Loop) {
        const tree = [
            {
                id: 'Pages',
                name: 'Pages',
                fullName: 'Pages',
                type: 'group',
                toggled: true,
                children: [] as object[],
            },
        ];

        loop.pages.forEach(page => {
            const jsonPages = {
                id: page.id,
                name: page.pageName,
                fullName: `${page.pageName} [${page.id}]`,
                type: 'page',
                obj: page,
                children: [] as object[],
            };

            page.questions.forEach(question => {
                jsonPages.children.push({
                    id: question.id,
                    name: question.name,
                    type: 'question',
                    fullName: `${question.name} [${question.id}]`,
                    obj: question,
                    children: this.qChildGroupControl(question.question, page,
questionToExclude),
                });
            });

            tree[0].children.push(jsonPages);
        });

        return tree;
```

```
    }

    public qChild(q: Question[], page: Page, isNested: boolean = false): any {
        if (!q || q[0] === null || q.length < 1) {
            return null;
        }
        return q.map((x: Question) => {
            return {
                id: x.id,
                ref: x.ref,
                name: x.name,
                fullName: `${x.name} [${x.id}]`,
                type: 'question',
                page,
                obj: x,
                isNested,
                children: this.qChild(x.question, page, isNested),
            };
        });
    }

    public qChildGroupControl(q: Question[], page: Page, questionToExclude: Question =
null): any {
        if (!q || q[0] === null) {
            return null;
        }
        return q.filter(x => x.type === 'GroupControl').length > 0
            ? q
                  .filter(x => x.type === 'GroupControl')
                  .map((x: Question) => {
                      return {
                          id: x.id,
                          name: x.name,
                          fullName: `${x.name} [${x.id}]`,
                          type: 'question',
                          obj: x,
                          children: this.qChildGroupControl(x.question, page,
questionToExclude),
                      };
                  })
            : null;
    }

    public findParentNode(tree: any, searchNode: any) {
        let parentNode: any;
        tree = tree[0] ? tree[0] : tree;
        tree.id === searchNode.ref || tree.id === searchNode.page ? (parentNode = tree) :
null;
        if (tree.children) {
```

```
            tree.children.forEach((item: any) => {
                if (item.id === searchNode.ref) {
                    parentNode = parentNode ? parentNode : item;
                } else {
                    parentNode = parentNode ? parentNode : this.findParentNode(item,
searchNode);
                }
            });
        }
        return parentNode;
    }

    public addNode(tree: any, addNode: any) {
        const parentNode = this.findParentNode(tree, addNode);
        parentNode.children ? null : (parentNode.children = []);
        const page = parentNode.page ? parentNode.page : parentNode.obj;
        const node = {
            id: addNode.id,
            name: addNode.name,
            fullName: `${addNode.name} [${addNode.id}]`,
            type: 'question',
            page,
            obj: addNode,
            ref: addNode.ref ? addNode.ref : addNode.page,
            children: this.qChild(addNode.question, page),
        };
        parentNode.children.push(node);
        return tree;
    }

    public getNode(tree: any, id: string) {
        let node: any;
        tree = tree[0] ? tree[0] : tree;
        if (tree.children) {
            tree.children.forEach((item: any) => {
                if (item.id === id) {
                    node = node ? node : item;
                } else {
                    node = node ? node : this.getNode(item, id);
                }
            });
        }
        return node;
    }

    public toggleParentNodes(tree: any, node: any) {
        let parentNode = this.findParentNode(tree, node);
        while (parentNode !== undefined) {
            parentNode.toggled = true;
```

```
            parentNode = this.findParentNode(tree, parentNode);
        }
    }

    public removeNode(tree: any, node: any) {
        const parentNode = this.findParentNode(tree, node);
        parentNode.children.splice(parentNode.children.findIndex((x: any) => x.id ===
node.id), 1);
    }
}
```

DOKUMENTĀRĀ LAPA

Kvalifikācijas darbs „Domēna specifiskās valodas redaktora izstrāde” izstrādāts Latvijas Universitātes Datorikas fakultātē.

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

Autors: ***Ralfs Cimermanis*** _____ **___.05.2019**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***M.dat. Valdis Vizulis*** _____ **___.05.2019.**

Recenzents: ***M.dat. Jānis Vempers***

Darbs iesniegts 27.05.2019.
Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2019. prot. Nr. _____

Komisijas sekretārs(-e): _____