

LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

**SAP SCREEN PERSONAS 3.0 CHROME
IZSTRĀDĀTĀJA RĪKU PAPLAŠINĀJUMA
IZSTRĀDE**

KVALIFIKĀCIJAS DARBS

Autors: Inga Pīre

Studenta apliecības Nr.: ip14047

Darba vadītājs: Dipl. mat. Imants Pops

RĪGA 2016

ANOTĀCIJA

SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājums ir programmatūras sastāvdaļa, kas ļauj pievienot papildus funkcionalitāti pārlūkprogrammai Google Chrome. Tas pievieno Google Chrome izstrādātāja rīkiem jaunu paneli “Personas”, nodrošina nepieciešamo datu iegūšanu no inspicējamā loga un to attēlošanu panelī.

Paplašinājums ir izstrādāts lietošanai kopā ar SAP Screen Personas 3.0, tā mērķis ir vienkāršot atklūdošanas un problēmu novēršanas procesus, attēlojot pašreizējo klienta lietojumprogrammas datu stāvokli skaidrā un viegli saprotamā veidā. Paplašinājums ir paredzēts konkrētam lietotāju lokam - SAP Screen Personas 3.0 izstrādātājiem.

Paplašinājums ir izstrādāts JavaScript programmēšanas valodā, lietojot jQuery bibliotēku, SAPUI5 rīkkopu, Chrome DevTools extensions API un uzdevumu pildītāju Grunt.

Atslēgvārdi: JavaScript, SAPUI5, Chrome DevTools paplašinājums, SAP Screen Personas 3.0

ABSTRACT

SAP Screen Personas 3.0 Chrome Developer Tools extension

SAP Screen Personas 3.0 Chrome Developer Tools extension is a software component, that allows to add an additional functionality to Google Chrome browser. Extension adds a new panel “Personas” to Google Chrome Developer Tools window, evaluates code in the inspected window to get the needed information and displays acquired information in the panel.

The goal of extension is to simplify SAP Screen Personas 3.0 debugging and troubleshooting process by presenting current client application state in clear and readable manner. The extension is meant for a specific user group – SAP Screen Personas 3.0 developers.

Extension is developed in JavaScript programming language using jQuery library, SAPUI5 toolkit, Chrome DevTools extensions API and task runner Grunt.

Keywords: JavaScript, SAPUI5, Chrome DevTools Extension, SAP Screen Personas 3.0

Satura rādītājs

VĀRDNĪCA	8
IEVADS	10
1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA	11
1.1 Ievads.....	11
1.1.1 Nolūks.....	11
1.1.2 Darbības sfēra	11
1.1.3 Definīcijas.....	11
1.1.4 Saistība ar citiem dokumentiem	11
1.1.5 Pārskats.....	11
1.2 Vispārējais apraksts	12
1.2.1 Esošā stāvokļa apraksts	12
1.2.2 Produkta perspektīva	12
1.2.3 Produkta funkcijas	13
1.2.4 Lietotāji.....	13
1.2.5 Vispārējie ierobežojumi.....	13
1.2.6 Pieņēmumi un atkarības.....	14
1.3 Funkcionālās prasības.....	14
1.3.1 Ievads.....	14
1.3.2 Modulis “Modeļu attēlošana ”	14
1.3.3 Modulis “Modeļa datu atjaunošana”	18
1.3.4 Modulis “Saziņa”	21
1.5 Ārējā saskarne.....	23
1.5.1 Lietotāja saskarne	23
1.5.2 Programmatūras saskarne	23
1.4 Nefunkcionālās prasības	24
1.4.1 Dokumentācija.....	24
1.4.2 Kļūdu pārvaldība	24

1.4.3 Instalējamība.....	24
2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS	25
2.1 Ievads.....	25
2.1.1 Nolūks.....	25
2.1.2 Darbības sfēra	25
2.1.3 Definīcijas.....	25
2.1.4 Saistība ar citiem dokumentiem	25
2.2. Dekompozīcijas apraksts	25
2.2.1 Trasējamības tabula	26
2.2.2 “SAPUI5 lietotne” uzbūve	28
2.2.3 “SAPUI5 lietotne” klašu dekompozīcija	28
2.2.4 “Chrome paplašinājums” uzbūve	33
2.2.5 "Chrome paplašinājums” klašu dekompozīcija	35
2.3 Nefunkcionālo prasību realizācija	36
2.4 Lietotāja saskarnes apraksts.....	36
2.4.1 Modeļu skats.....	37
2.4.2 Paziņojumu parādīšana	39
3. TESTĒŠANAS DOKUMENTĀCIJA.....	40
3.1 Ievads.....	40
3.1.1 Nolūks.....	40
3.1.2 Definīcijas.....	40
3.1.3 Saistība ar citiem dokumentiem	40
3.1.4 Testēšanas metodika	40
3.2 Trasējamības tabula	41
3.3 Testēšanas žurnāls	42
3.4 Testēšanas kļūdu apstrāde	48
4. PROJEKTA ORGANIZĀCIJA.....	49
5. KVALITĀTES NODROŠINĀŠANA.....	50

6. KONFIGURĀCIJAS PĀRVALDĪBA	51
7. DARBIETILPĪBAS NOVĒRTĒJUMS	52
7.1 Darbietilpības novērtējums - pirms un pēc	52
7.2 Eksperta metode	53
8. IZMANTOTĀS TEHNOLOĢIJAS UN RĪKI	54
8.1 SAPUI5 rīkkopa	54
8.2 Uzdevumpildītājs Grunt	55
9. PROGRAMMATŪRAS KODS	56
9.1 Kontrollera klases “Models” metode “onReloadAll”	56
9.2 Kontrolleru klases “Models” metode “loadAllModels”	57
9.3 Klases “ChromeExtensionDataProvider” metode “getModelNames”	57
9.4 Datne “getModelNames.js”	58
SECINĀJUMI	60
IZMANTOTĀ LITERATŪRA	61
PIELIKUMI.....	62
1. Metodes “parse” ievaddatu un izvaddatu piemērs.....	62
2. Skatam piesaistītā modeļa uzbūve	63
3. Prasības “1.3.3.2 Visu modeļu atjaunināšana” realizācijas grafisks attēlojums.....	64
4. Prasības “1.3.3.1 Modeļa atjaunināšana” realizācijas grafisks attēlojums	65
5. Prasības “1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski realizācijas grafisks attēlojums	66
6. SonarQube ekrānu uzņēmumi	67
7. Atlassian JIRA uzdevumu un problēmu pieteikumi.....	68
8. JSDoc.....	69
8.1 Kontrolleru klase “Models”	69
8.2 Kontroles klase “XMLViewer”	72
8.3 Kontroles klase “JSONViewer”	72
8.4 Klase “XMLParser”.....	73

8.5 Klase “ChromeExtensionDataProvider”	73
8.6 Klase “MessageHandler”.....	74
9. User guide.....	75
1. Introduction	75
2. Overview	75
3. Tutorial	76

VĀRDNĪCA

API (Application programming interface) - lietojumprogrammas saskarne jeb iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums.

Atlassian JIRA – sistēma, kas paredzēta projekta uzdevumu un problēmu pieteikumu apstrādei.

COCOMO (Constructive Cost Model) - metode, ar kuras palīdzību novērtē lietotājam piegādājamās sistēmas izveides darbietilpību no lietotāja viedokļa.

COCOMO II (The Constructive Cost Model II) - jaunāka versija COCOMO metodei, kas labāk piemērota mūsdienīgu programmatūras projektu apjomu vērtēšanai.

CSS (Cascading Style Sheets) - īpaša stila lapas valoda, ko lieto, lai aprakstītu izskatu iezīmēšanas valodā veidotiem dokumentiem.

DOM (Document Object Model) - dokumenta objekta modelis.

Google Chrome - interneta pārlūkprogramma.

Grunt – uzdevumu pildītājs, kas automatizē atkārtotu uzdevumu pildīšanu.

GUI (Graphical User Interface) – grafiskā lietotāja saskarne ir lietotāja saskarnes veids, kas ļauj lietotājam sadarboties ar datoru caur grafiskām ikonām un vizuāliem indikatoriem.

HTML (Hypertext Markup Language) - standartizēta teksta datņu iezīmēšanas valoda, kas paredzēta tīmekļa vietņu veidošanai.

Inspicējamā loga kodolā reģistrētie modeļi –

1. **Inspicējamais logs** – pārlūkprogrammas Google Chrome cilnē atvērtais logs (konkrēti SAP Screen Personas 3.0 Client), ar kuru notiek komunikācija un no kura tiek iegūti dati;
2. **Kodols** – sap.ui.core.Core objekts;
3. **Reģistrētie modeļi** - sap.ui.model.Model objekti.

JavaScript - skriptu valoda, kas balstīta uz prototipu koncepta, izmanto tīmekļa vietnēs izpildīšanai uz klienta datora.

Jenkins - nepārtrauktās integrācijas rīks, kas ļauj automātiski veikt būvējumus pēc izstrādātāju veiktajām koda izmaiņām.

jQuery - JavaScript bibliotēka, izveidota, lai vienkāršotu HTML skriptēšanu.

JSDoc – iezīmēšanas valoda, lietota lai komentētu JavaScript pirmkoda datnes.

JSON (JavaScript Object Notation) — JavaScript Objektu Notācija - datu apmaiņas formāts.

LESS – CSS priekšapstrādātājs.

MVC (Model – View - Controller) - projektēšanas šablons „Modelis – Skats - Kontrolleris”.

Objekts - datu kopums kopā ar tām piesaistītajām darbībām.

PPS - programmatūras prasību specifikācija.

PPA – programmatūras projektējuma apraksts

SAP Screen Personas 3.0 – uz tīmekļa pārlūkprogrammas bāzēta atveidošanas programmatūra klasiskiem SAP Dynpro ekrāniem. Šis produkts nodrošina lietotāju līmeņa personalizācijas iespējas, kas ļauj IT profesionāļiem un gala lietotājiem ātri un viegli vienkāršot biznesa lietotņu ekrānus, uzlabojot izskatu, gala lietotāja produktivitāti un SAP veiktspēju.

SAPUI5 - lietotāja saskarnes izstrādes rīkkopa priekš HTML5.

SonarQube - platforma koda kvalitātes pārvaldīšanai.

SVN – programmatūras versiju izveides un pārskatīšanas sistēma.

UI (User Interface) – lietotāja saskarne, visu programmā vai datorā paredzēto līdzekļu kopums, kas nosaka, kā lietotājs var sadarboties ar datoru.

XML (eXtensible Markup Language) - paplašināmā iezīmēšanas valoda, kas paredzēta visdažādāko datu aprakstīšanai.

IEVADS

Kvalifikācijas darba mērķis ir izstrādāt Chrome izstrādātāja rīku paplašinājumu, kas paredzēts konkrēti priekš SAP Screen Personas 3.0.

SAP ir uzņēmums, kas nodarbojas ar biznesa programmatūras izstrādi un apkalpošanu, viens no šī uzņēmuma produktiem ir SAP Screen Personas 3.0. Šī produkta galvenais mērķis ir nodrošināt uzņēmuma personālam viegli lietojamu un personalizētu sistēmas pārvaldības vidi SAP sistēmā. SAP Screen Personas 3.0 lietotājam ļauj pārveidot gan ekrāna funkcionalitāti, gan vizuālo izskatu, zinot tikai dažus programmēšanas konceptus.

SAP Screen Personas 3.0 lietotāju ekrāni ir balstīti uz modeļiem, tāpēc, lai izstrādātājiem būtu vieglāk atrast kļūdas un novērst tās, ir nepieciešams attēlot pašreizējo klienta lietotnes stāvokli – parādīt visus lietotnes kodolā reģistrētos XML un JSON modeļus saprotamā veidā ar formatējumu, kā arī izmaiņu gadījumā automātiski tos atjaunināt. Kā piemēram, visas lietotāja veiktās izmaiņas personalizētajam ekrānam (Flavor), tiek glabātas vienā no modeļiem, redzot šo modeli ir iespējams noteikt, vai no aizmugursistēmas nāk pareizi dati un vai lietotāja saskarne atbilst modelim.

Paplašinājums ir veidots pārlūkprogrammai Google Chrome. Tā izstrādē ir izmantots Chrome DevTools Extensions API, lai nodrošinātu komunikāciju starp inspicējamo logu un paplašinājumu. Viena no paplašinājuma galvenajām funkcijām ir jauna paneļa "Personas" pievienošana Chrome izstrādātāja rīku logam, tajā tiek attēloti visi iegūtie XML un JSON modeļi ar formatējumu. Lai uzlabotu modeļu pārskatāmību, tie ir organizēti cilnēs. Panelis sastāv no HTML datnes, kurā ir iekļauti arī citi resursi (JavaScript, CSS). "Personas" paneļa lietotāja saskarne ir izstrādāta izmantojot SAPUI5 rīkkopu.

Kvalifikācijas darba ietvaros tika izstrādāta arī programmatūras - SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma - dokumentācija, kas ietver programmatūras prasību specifikāciju, programmatūras projektējuma aprakstu, testēšanas dokumentāciju, projekta organizāciju, konfigurācijas pārvaldību, kvalitātes nodrošināšanu, darbietilpības novērtējumu un programmatūras koda paraugu, kā arī lietotāja dokumentāciju angļu valodā.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1 Ievads

1.1.1 Nolūks

Šī programmatūras prasību specifikācija (PPS) ir paredzēta un izstrādāta SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma prasību definēšanai un to aprakstīšanai. Dokuments skaidri un viennozīmīgi definē produkta prasības. Tas ir paredzēts paplašinājuma izstrādātājam, lai definētu prasības un funkcijas tālākai paplašinājuma izstrādei, kā arī produkta pasūtītājam, lai precīzi definētu produkta prasības un apstiprinātu produkta tālāko izstrādi.

1.1.2 Darbības sfēra

Izstrādātā programmatūra – paplašinājums – ir paredzēts kā palīgs SAP Screen Personas 3.0 izstrādātājiem. Programmatūra iegūst pašreizējo klienta lietotnes stāvokli raksturojošus datus – JSON un XML modeļus – un attēlo tos vienkāršā un saprotamā veidā. Paplašinājums ir izstrādāts priekš pārlūkprogrammas Google Chrome, kur tas pievieno jaunu paneli “Personas” izstrādātāja rīkiem. Panelī tiek attēloti visi inspicējamā loga, konkrētāk SAP Screen Personas 3.0 Client, kodolā reģistrētie modeļi JSON un XML formātā.

1.1.3 Definīcijas

Skatīt “*VĀRDNĪCA*” (8.lpp.).

1.1.4 Saistība ar citiem dokumentiem

Programmatūras prasību specifikācijas izstrādē un noformēšanā tika ievērotas standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis” prasības.

1.1.5 Pārskats

Šī programmatūras prasību specifikācija sastāv no 5 apakšnodaļām:

1. Ievads. Sniedz informāciju par programmatūras prasību specifikācijas nolūku un darbības sfēru, kā arī iepazīstina ar darba uzbūvi.
2. Vispārējais apraksts. Satur vispārēju informāciju par programmatūru, tas ir, tās esošā stāvokļa aprakstu, produkta perspektīvām, lietotājiem un galvenajām funkcijām. Tiek sniegta informācija arī par sistēmas ierobežojumiem.

3. Funkcionālās prasības. Sniedz informāciju par programmatūras nodrošinātajām funkcijām.
4. Nefunkcionālās prasības. Šajā nodaļā ir apkopota informācija par specifiskajām prasībām, kas ir izvirzītas, piemēram, pret programmatūras instalējamību, dokumentāciju un kļūdu pārvaldību.
5. Ārējā saskarne. Tiek specificēta informācija par raksturiezīmēm, kuras jāatbalsta programmatūrai attiecībā uz katru produkta saskarni.

1.2 Vispārējais apraksts

1.2.1 Esošā stāvokļa apraksts

Šobrīd vēl nepastāv paplašinājums ar šādu funkcionalitāti, kas paredzēts konkrēti SAP Screen Personas 3.0 izstrādātājiem, attiecīgi šī paplašinājuma izstrāde ir oriģināla un tirgū vēl neesoša. Šis produkts vienkāršotu un atvieglotu SAP Screen Personas 3.0 izstrādātāju darbu.

Šis paplašinājums ļauj:

- Iegūt nepieciešamos datus vieglāk un ātrāk;
- Samazināt izstrādes laiku, jo nav nepieciešams daudzas reizes veikt atkārtotas darbības;
- Attēlot pašreizējo klienta lietojumprogrammas datu stāvokli vienkāršā un saprotamā veidā.

Paplašinājumu iespējams izmantot kā pamata bāzi, un to būtu iespējams pielāgot izmantošanai citās pārlūkprogrammās, ne tikai Google Chrome.

1.2.2 Produkta perspektīva

Izstrādātā programmatūra ir atkarīga no kādas citas jau pastāvošas programmatūras, tā ir veidota kā paplašinājums pārlūkprogrammai Google Chrome. Paplašinājums nodrošina iespēju pievienot papildus funkcionalitāti Google Chrome izstrādātāja rīkiem izmantojot Chrome DevTools Extensions API. Kā arī paplašinājums ir atkarīgs no SAP Screen Personas 3.0 Client, kam šis paplašinājums ir izstrādāts un ar ko notiek komunikācija arī izmantojot Chrome DevTools Extensions API.

1.2.3 Produkta funkcijas

Visas programmatūras funkcijas ir strukturētas moduļu veidā. Katrs no moduļiem piedāvā šādu funkcionalitāti:

- Modulis “Modeļu attēlošana” - nodrošina nepieciešamās funkcijas formatētu JSON un XML modeļu attēlošanai ar sintakses iezīmēšanu un ar iespējām izvērst un sakļaut sadaļas, kā arī visu modeļu saraksta apskatīšanu. Šis modulis ietver šādas funkcijas:
 - Visu modeļu saraksta attēlošana;
 - XML modeļa attēlošana;
 - JSON modeļa attēlošana;
 - Pēdējo ielādēto modeļu saraksta attēlošana sānjoslā.
- Modulis “Modeļu datu atjaunošana” - nodrošina modeļu datu atjaunošanu, gan pēc lietotāja iniciatīvas, gan automātiski, ja kādā modelī ir notikušas izmaiņas. Šis modulis ietver šādas funkcijas:
 - Modeļa atjaunināšana;
 - Visu modeļu atjaunināšana;
 - Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski.
- Modulis “Saziņa” - šī moduļa funkcijas nodrošina iespēju paplašinājumam sazināties ar inspicējamo logu.
 - Savienojuma izveidošana starp paplašinājumu un inspicējamo logu;
 - Savienojuma pārtraukšana starp paplašinājumu un inspicējamo logu.

1.2.4 Lietotāji

Paplašinājums ir paredzēts konkrētam lietotāju lokam - SAP Screen Personas 3.0 izstrādātājiem.

1.2.5 Vispārējie ierobežojumi

Paplašinājumam ir jāatbalsta Google Chrome pēdējās versijas (sākot ar Chrome 44.0).

Paplašinājums ir izstrādāts kā papildinājums jau esošai sistēmai, tāpēc tam jāizpildās vispārējās sistēmas prasībām:

- Produktam jābūt izstrādātam angļu valodā;
- Programmatūras kodam jābūt izstrādātām saistībā ar JavaScript koda vadlīnijām.

1.2.6 Pieņēmumi un atkarības

Veiksmīgai programmatūras darbībai, tiek pieņemts, ka lietotājs strādās ar kādu no pārlūkprogrammas Google Chrome pēdējām versijām (sākot ar Chrome 44.0), kurai ir aktivizēta JavaScript programmēšanas valodas skriptu izpilde. Kā arī tiek pieņemts, ka lietotājam ir pieeja SAP Screen Personas 3.0 Client.

1.3 Funkcionālās prasības

1.3.1 Ievads

Sistēma ir sadalīta 3 lielākos moduļos, kur katrs no moduļiem nodrošina noteiktu funkcionalitāti. Katras prasības identifikatoru veido nodaļas numurs un prasības nosaukums (x.y.z.u prasības nosaukums), piemēram “1.3.2.1 Visu modeļu saraksta attēlošana”.

1.3.2 Modulis “Modeļu attēlošana ”

Šī moduļa funkcijas nodrošina iespēju attēlot formatētus JSON un XML modeļus ar sintakses iezīmēšanu un ar iespējām izvērst un sakļaut sadaļas, kā arī visu modeļu saraksta apskatīšanu.

1.3.2.1 Visu modeļu saraksta attēlošana

tabula 1.1

Ievads
Šī funkcija nodrošina iespēju attēlot sarakstu cilņu veidā ar visiem inspicējamā loga kodolā reģistrētajiem JSON un XML modeļiem. Cilņu skaits iepriekš nav zināms, tas ir atkarīgs no modeļu skaita, kas tiek dinamiski iegūts.
Ievaddati
Ievaddatu nav, jo funkcija tiek izsaukta automātiski pie paneļa “Personas” atvēršanas jeb skata inicializācijas.
Apstrāde
Skatam tiek piesaistīts modelis, kas satur visu inspicējamā logā reģistrēto JSON un XML modeļu datus, nosaukumus un tipus, kā arī pēdējo modeļa pārlādēšanas laiku.

<p>Cilnes tiek ģenerētas no skatam piesaistītā modeļa datiem. Cilņu galvenēm (jeb cilņu nosaukumiem) tiek piesaistīti modeļu nosaukumi, kamēr cilņu saturam tiek piesaistīti attiecīgā modeļa dati, modeļa tips un modeļa pārlādēšanas laiks.</p> <p>Cilnēm piesaistīto datu tālāku apstrādi atkarībā no modeļa tipa skatīt funkcijās “1.3.2.2 XML modeļa attēlošana” un “1.3.2.3 JSON modeļa attēlošana”.</p> <p>Ja skatam piesaistītajā modelī dati tiek izmainīti, automātiski tiek atjaunināta arī lietotāja saskarne.</p> <p>Ja skatam piesaistītajā modelī nav datu - parāda “No data”.</p>
Izvaddati
Skatā tiek attēlots saraksts ar visiem JSON un XML modeļiem cilņu veidā.
Kļūdu ziņojumi
-

1.3.2.2 XML modeļa attēlošana

tabula 1.2

Ievads
Šī funkcija nodrošina iespēju attēlot formatētu XML modeli ar sintakses iezīmēšanu un ar iespējām izvērst un sakļaut sadaļas.
Ievaddati
<p>Tiek izvēlēta cilne nospiežot uz modeļa nosaukuma un kā ievaddati tiek padoti:</p> <ol style="list-style-type: none"> 1. Cilnei piesaistītie XML modeļa dati (JSON formāta dati, kas apraksta XML modeļa uzbūvi, obligāts) (cilnei dati tiek piesaistīti funkcijā “1.3.2.1 Visu modeļu saraksta attēlošana”).
Apstrāde
XML modeļa dati uz ekrāna tiek attēloti ar šādu formatējumu:

<ul style="list-style-type: none"> • XML elementu mezgli bez bērniem tiek attēloti vienā līnijā; • XML elementu mezgli ar bērniem tiek attēloti 2+x (x - dotā elementa mezgla bērnu skaits) līnijās, kur pirmajā līnijā ir atverošais tags ar atribūtiem un atribūtu vērtībām, nākamajās līnijas atrodas bērnu mezgli un pēdējā līnijā ir aizverošais tags; • XML elementu mezglu bērni tiek attēloti ar atkāpi no vecāka līnijas sākuma; • XML elementi, elementu vērtības atribūti un atribūtu vērtības ir iezīmētas atšķirīgās krāsās; • XML elementu mezgliem ar bērniem, ir iespējams bērnus izvērst (parādīt) vai sakļaut (paslēpt); <p>Ja cilnei piesaistītie XML modeļa dati tiek izmainīti, automātiski tiek atjaunināta arī lietotāja saskarne.</p>
Izvaddati
Lietotājam tiek attēlots formatēts XML modelis.
Kļūdu ziņojumi
-

1.3.2.3 JSON modeļa attēlošana

tabula 1.3

Ievads
Šī funkcija nodrošina iespēju attēlot formatētu JSON modeli ar sintakses iezīmēšanu un ar iespējām izvērst un sakļaut sadaļas.
Ievaddati
<p>Tiek izvēlēta cilne nospiežot uz modeļa nosaukuma un kā ievaddati tiek padoti:</p> <ol style="list-style-type: none"> 1. Cilnei piesaistītie JSON modeļa dati (JSON formāta dati, obligāts) (cilnei dati tiek piesaistīti funkcijā “1.3.2.1 Visu modeļu saraksta attēlošana”.
Apstrāde

<p>JSON modeļa dati uz ekrāna tiek attēloti ar šādu formatējumu:</p> <ul style="list-style-type: none"> • Objekts tiek iekļauts figūriekavās, objekta dati tiek attēloti jaunā rindā ar atkāpi no figūriekavu sākuma. Aiz katras objekta atslēgas seko kols, aiz kura seko atslēgas vērtība. Objekta vērtības atdala ar komatu. Katru nākamo objektu atdala ar komatu; • Masīvs tiek iekļauts kvadrātiem, katrs masīva elements tiek attēlots jaunā rindā ar atkāpi no kvadrātiem sākuma. Masīva elementi tiek atdalīti ar komatu; • Objekta un masīva elementus ir iespējams izvērst (parādīt) vai sakļaut (paslēpt); • Sakļaujot kādu sadaļu tiek parādīts sakļauto elementu skaits; • Veseli skaitļi, decimālskaitļi un būla vērtības tiek attēlotas atšķirīgā krāsā no simbolu virknēm. <p>Ja cilnei piesaistītie JSON modeļa dati tiek izmainīti, automātiski tiek atjaunināta arī lietotāja saskarne.</p>
Izvaddati
Lietotājam tiek attēlots formatēts JSON modelis.
Kļūdu ziņojumi
-

1.3.2.4 Pēdējo ielādēto modeļu saraksta attēlošana sānjoslā

tabula 1.4

Ievads
Šī funkcija nodrošina iespēju parādīt vai paslēpt sarakstu ar 15 pēdējo ielādēto modeļu datiem – to nosaukumiem un ielādēšanas laiku. Kā arī saraksta automātisku atjaunināšanu, ielādējot kādu jaunu modeli.
Ievaddati
<p>Pēc pogas “Show last loaded” vai “Hide last loaded” nospiešanas kā ievaddati tiek padoti:</p> <ol style="list-style-type: none"> 1. Notikuma objekts (objekts, obligāts) - satur informāciju par notikuma avotu - pogas

kontroli un tās parametriem (piemēram, tekstu, redzamību, ikonu utt.), kas ir uzstādīti vai sasaistīti ar modeli.
Apstrāde
<p>Pēc pogas nospiešanas tiek noteikts pēc kontroles parametriem, vai sānjosla ir paslēpta vai redzama.</p> <p>Ja ir redzama, sānjosla tiek paslēpta nomainot vērtību kontrolei piesaistītajā modelī.</p> <p>Ja nav redzama, sānjosla tiek parādīta nomainot vērtību kontrolei piesaistītajā modelī.</p> <p>Katru reizi, kad tiek ielādēts kāds modelis, saraksts sānjoslā tiek papildināts, ja sarakstā ir vairāk par 15 ierakstiem – pēdējos dzēš ārā. Veicot izmaiņas sarakstā, saskarne tiek automātiski atjaunināta.</p> <p>Ja sarakstā nav neviena ieraksta, tiek parādīts teksts “No data”.</p>
Izvaddati
Skata sānjoslā tiek attēlots vai paslēpts saraksts ar 15 pēdējiem ielādētajiem modeļiem.
Kļūdu ziņojumi
-

1.3.3 Modulis “Modeļa datu atjaunošana”

Šī moduļa funkcijas nodrošina modeļu datu atjaunošanu, gan pēc lietotāja iniciatīvas, gan automātiski, ja kādā modelī ir notikušas izmaiņas.

1.3.3.1 Modeļa atjaunināšana

tabula 1.5

Ievads
Šī funkcija nodrošina iespēju lietotājam pārlādēt kādu modeli manuāli – tiek atjaunoti modeļa dati un pārlādēšanas laiks.
Ievaddati

<p>Pēc pogas “Reload” nospiešanas kā ievaddati tiek padoti:</p> <ol style="list-style-type: none"> 1. Notikuma objekts (objekts, obligāts) - satur informāciju par notikuma avotu - pogas kontroli un tās parametriem (piemēram, tekstu, redzamību, ikonu utt.), kas ir uzstādīti vai sasaistīti ar modeli.
<p>Apstrāde</p>
<p>Pēc pogas “Reload” nospiešanas tiek pārbaudīts vai inspicētajā logā ir pieejama SAPUI5 rīkkopa.</p> <p>Ja nav, tiek izdots kļūdu paziņojums Nr. 1.</p> <p>Ja ir, no konteksta (piesaistītajiem datiem) tiek atrasts modelis, kuram tika izsaukta šī funkcija. Tālāk, izmantojot chrome.devtools.inspectedWindow API, no inspicējamā loga tiek iegūti šī modeļa dati.</p> <p>Ja šis modelis ir deklarēts, bet tam nav uzstādīti dati, tiek izdots kļūdu paziņojums Nr. 2.</p> <p>Ja modelī ir kļūda formātā (atkarībā no tipa neatbilst JSON vai XML formātam) un to nav iespējams noparsēt, tiek izdots kļūdu paziņojums Nr. 3.</p> <p>Ja kļūdas netiek konstatētas, iegūtie dati tiek saglabāti skatam piesaistītajā modelī. Izmaiņas modelī izsauc automātisku saskarnes atjaunināšanu.</p>
<p>Izvaddati</p>
<p>Skatā tiek attēlots atjaunināts modelis - tā dati un ielādēšanas laiks.</p>
<p>Kļūdu ziņojumi</p>
<ol style="list-style-type: none"> 1. Couldn't load model <modeļa nosaukums>, because SAPUI5 isn't available! 2. Couldn't load model <modeļa nosaukums>, because there is no data set to this model! 3. Couldn't load model <modeļa nosaukums>, because there is an error in <modeļa tips> format!

1.3.3.2 Visu modeļu atjaunināšana

tabula 1.6

<p>Ievads</p>

Šī funkcija nodrošina iespēju lietotājam atjaunot modeļu sarakstu un visu modeļu datus, kā arī nodrošina modeļu saraksta un to datu pirmreizēju iegūšanu.
Ievaddati
Ievaddatu nav, jo funkcija tiek izsaukta pie Chrome izstrādātāja rīku paneļa “Personas” atvēršanas, kad visi modeļi tiek ielādēti pirmo reizi, vai pogas “Reload all models” nospiešanas.
Apstrāde
<p>Pēc paneļa “Personas” atvēršanas vai pogas “Reload all models” nospiešanas tiek pārbaudīts vai inspicētajā logā ir pieejama SAPUI5 rīkkopa.</p> <p>Ja nav, tiek izdots kļūdu paziņojums Nr. 1.</p> <p>Ja ir, izmantojot chrome.devtools.inspectedWindow API, tiek iegūti dati par inspicētajā logā reģistrētajiem JSON un XML modeļiem – nosaukums, modeļa tips (JSON vai XML) un modeļa dati.</p> <p>Ja kāds no modeļiem ir deklarēts, bet tam nav uzstādīti dati, tiek izdots kļūdu paziņojums Nr. 2.</p> <p>Ja kādā no XML vai JSON modeļiem ir kļūda formātā un to nav iespējams noparsēt, tiek izdots kļūdu paziņojums Nr. 3.</p> <p>Ja kļūdas netiek konstatētas, katra modeļa iegūtie dati tiek saglabāti modelī, kas piesaistīts skatam. Izmaiņas modelī izsauc automātisku saskarnes atjaunināšanu.</p>
Izvaddati
Skatā tiek attēlots atjaunināts saraksts ar modeļiem, to datiem un ielādēšanas laiku.
Kļūdu ziņojumi
<ol style="list-style-type: none"> 1. Couldn't load models, because SAPUI5 isn't available! 2. Couldn't load model <modeļa nosaukums>, because there is no data set to this model! 3. Couldn't load model <modeļa nosaukums>, because there is an error in <modeļa tips> format!

1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski

tabula 1.7

Ievads
Šī funkcija nodrošina iespēju uzraudzīt izmaiņas modeļos un izmaiņu gadījumā automātiski atjaunināt modeļa datus un saskarni.
Ievaddati
Ievaddatu nav, jo funkcija tiek izsaukta pie Chrome izstrādātāja rīku paneļa “Personas” atvēršanas, kad visi modeļi tiek ielādēti pirmo reizi.
Apstrāde
<p>Izmantojot chrome.devtools.inspectedWindow API, inspicētajā logā tiek izpildīts skripts, kas XML un JSON modeļiem piesaista notikumu uztvērēju un izmaiņu gadījumā nodod ziņas objektu uz paplašinājumu.</p> <p>Paplašinājumā saņemot ziņas objektu, šim modelim tiek veikta funkcijā “1.3.3.1 Modeļa atjaunināšana” definētās darbības, ar vienu atšķirību, ka modeļa dati – nosaukums un tips - netiek iegūti no konteksta (no piesaistītajiem datiem), bet gan tiek iegūti no saņemtā ziņas objekta. Arī kļūdu ziņojumi saglabājas tie paši.</p>
Izvaddati
-
Kļūdu ziņojumi
-

1.3.4 Modulis “Saziņa”

Šī moduļa funkcijas nodrošina iespēju paplašinājumam sazināties ar inspicējamo logu.

1.3.4.1 Savienojuma izveidošana starp paplašinājumu un inspicējamo logu

tabula 1.8

Ievads
Šī funkcija nodrošina savienojuma izveidošanu starp paplašinājumu un inspicējamo logu.

Ievaddati
<p>Šī funkcija tiek izsaukta lietotājam atverot paneli “Personas” un tai tiek padots ziņas objekts, kurā ir nepieciešamie dati, lai izveidotu savienojumu:</p> <ol style="list-style-type: none"> 1. ziņas nosaukums, kas nosaka, ka notiek jauna savienojuma inicializācija (simbolu virkne, iespējamās vērtības: “init”, obligāts); 2. attiecīgās cilnes ID (vesels skaitlis, obligāts).
Apstrāde
<p>Tiek veiktas pārbaudes ziņas datiem, vai tie atbilst prasībām, kas nepieciešami, lai izveidotu jaunu savienojumu.</p> <p>Ja prasības izpildās, tad pārbauda, vai savienojums jau neeksistē esošo savienojumu sarakstā.</p> <p>Ja savienojums neeksistē esošo savienojumu sarakstā, tad tas tiek pievienots.</p> <p>Ja prasības neizpildās vai savienojums jau eksistē esošo savienojumu sarakstā, ziņa tiek ignorēta.</p> <p>Savienojumam tiek pievienoti divi notikumu uztvērēji - ziņu uztveršanai no inspicētā loga un savienojuma pārtraukuma uztveršanai.</p>
Izvaddati
-
Kļūdu ziņojumi
-

1.3.4.2 Savienojuma pārtraukšana starp paplašinājumu un inspicējamo logu.

tabula 1.9

Ievads
Šī funkcija nodrošina savienojuma pārtraukšanu starp paplašinājumu un inspicējamo logu.
Ievaddati

Šī funkcija tiek izsaukta lietotājam aizverot paneli “Personas” un no notikumu uztvērēja tai tiek padoti dati attiecīgās cilnes identificēšanai: 1. attiecīgās cilnes ID (vesels skaitlis, obligāts).
Apstrāde
Savienojumam ar attiecīgās cilnes ID tiek noņemti visi notikumu uztvērēji, savienojums tiek dzēsts no visu savienojumu saraksta.
Izvaddati
-
Kļūdu ziņojumi
-

1.5 Ārējā saskarne

1.5.1 Lietotāja saskarne

Lietotāja saskarne ir jārealizē ar SAPUI5 lietotnes palīdzību. Tai ir jābūt viegli saprotamai – intuitīvai. Navigācijai starp XML un JSON modeļiem jābūt realizētai ar cilņu joslu – katram modelim jābūt savai cilnei, kurā tiek attēlots modeļa nosaukums un modeļa dati. Kļūdu, brīdinājumu, veiksmes un informācijas paziņojumi ir jāattēlo uznirstošajā izvēlnē. Skatā nevajag rādīt neko lieku, kontrolēm ir jābūt izkārtotām ērtā un viegli pārskatāmā veidā. Pēc iespējas vairāk ekrāna vietas jāvelta XML un JSON modeļu attēlošanai, lai tos būtu vieglāk pārskatīt. Datu attēlošanai, kontroļu paslēpšanai, parādīšanai vai kādu izmaiņu veikšanai lietotāja saskarnē ir jāizmanto modeļu piesaiste skatam un kontrolēm.

1.5.2 Programmatūras saskarne

Paplašinājums izmanto Google Chrome pārlūkprogrammu (sākot ar Chrome 44.0), paplašinājuma pievienošana pārlūkprogrammai tiek nodrošinātā izmantojot Chrome DevTools Extensions API.

1.4 Nefunkcionālās prasības

1.4.1 Dokumentācija

1.4.1.1 JSDoc

Pirmkodam ir jābūt labi komentētam JSDoc iezīmēšanas valodā, lai no tā varētu ģenerēt automātisko dokumentāciju.

1.4.1.2 Lietotāja dokumentācija

Programmatūras piegādāšanas brīdī ir jābūt izstrādātai un pieejamai lietotāja dokumentācijai angļu valodā.

1.4.2 Kļūdu pārvaldība

1.4.2.1 Problēmu un uzdevumu pārvaldība

Programmatūras izstrādes laikā jāizmanto projekta problēmu un uzdevumu pārvaldības rīks, lai tiktu korekti reģistrēti veicamie uzdevumi un apstrādājamās problēmas.

1.4.3 Instalējamība

1.4.3.1 Instalēšana un atinstalēšana

Lietotājam ir jāspēj instalēt, atinstalēt un vadīt programmatūru izmantojot tikai lietotāja dokumentāciju.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1 Ievads

2.1.1 Nolūks

Šis programmatūras projektējuma apraksts (PPA) ir SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma attēlojums, kas nodrošina precīzu informāciju programmatūras plānošanai, analīzei un implementēšanai. Dokuments parāda kā programmatūra ir strukturēta, lai apmierinātu prasības, kas identificētas programmatūras prasību specifikācijā. Tas ir paredzēts programmatūras izstrādātājiem.

2.1.2 Darbības sfēra

Izstrādātā programmatūra – paplašinājums – ir paredzēts kā palīgs SAP Screen Personas 3.0 izstrādātājiem. Paplašinājums ir izstrādāts priekš pārlūkprogrammas Google Chrome, kur tas pievieno jaunu paneli “Personas” izstrādātāja rīkiem. Panelī tiek attēloti visi inspicējamā loga, konkrētāk SAP Screen Personas 3.0 Client, kodolā reģistrētie modeļi JSON un XML formātā, kas raksturo pašreizējo klienta lietotnes datu stāvokli.

2.1.3 Definīcijas

Skatīt “*VĀRDNĪCA*” (8.lpp.).

2.1.4 Saistība ar citiem dokumentiem

Programmatūras projektējuma apraksta izstrādē un noformēšanā tiek ievēroti standarta LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai” norādījumi. Kā arī dokuments ir izstrādāts balstoties uz SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma programmatūras prasību specifikācijā izvirzītajām prasībām.

2.2. Dekompozīcijas apraksts

Izstrādātā programmatūra ir sadalīta divās daļās – “SAPUI5 lietotne”, kas ir izstrādāta izmantojot MVC pieeju un “Chrome paplašinājums”, kas ir izstrādāts izmantojot Chrome DevTools Extension API. Programmatūras daļas “SAPUI5 lietotne” klases un metodes ir galvenokārt atbildīgas par datu apstrādi un attēlošanu, kamēr “Chrome paplašinājums” klases un metodes, kā arī skripti saziņas nodrošināšanai starp inspicējamo logu un paplašinājumu, ir atbildīgi par datu ieguvu. Šīs daļas ir neatkarīgas viena no otras un nodrošina iespēju pielāgot

izstrādāto paplašinājumu arī citai videi. Līdz ar to, ja datu ieguve tiks pielāgota citai videi, datu apstrādes un attēlošanas funkcijas vel aizvien strādās. Kā, piemēram, pārveidojot datu ieguves daļu, izstrādātā programmatūra varētu strādāt arī kā parasta tīmekļa vietne statistiku XML un JSON modeļu attēlošanai.

Dekompozīcijas apraksts satur:

- trasējamības tabulu (sk. “2.2.1 *Trasējamības tabula*”), kurā attēlots kā visas PPS izvirzītās prasības ir realizētas – tiek norādīti attiecīgo klašu un metožu nosaukumi katrai prasībai;
- katras programmatūras daļas aprakstu – katrai daļai tiek paskaidrota uzbūve (sk. “2.2.2 *“SAPUI5 lietotne” uzbūve*” un “2.2.4 *“Chrome paplašinājums” uzbūve*”) un aprakstīta klašu dekompozīcija (sk. “2.2.3 *“SAPUI5 lietotne” klašu dekompozīcija*” un “2.2.5 *“Chrome paplašinājums” klašu dekompozīcija*”);
- lai uzzinātu sīkāk par programmatūras izstrādē izmantotajām tehnoloģijām, skatīt “8. *Izmantotās tehnoloģijas un Rīki*”.
- ar sīkāku klašu metožu aprakstu var iepazīties pielikumā “8. *JSDoc*” - automātiski ģenerētajā dokumentācijā.

2.2.1 Trasējamības tabula

tabula 2.1

PPS prasības identifikators	PPA klases nosaukums un metodes identifikators vai cita veida vienuma identifikators
1.3.2.1 Visu modeļu saraksta attēlošana	Kontrollera klases “Models” metodes: <ul style="list-style-type: none"> • 2.2.3.3.1 onInit; • 2.2.3.3.2 modelListFactory.
1.3.2.2 XML modeļa attēlošana	Kontroles klases “XMLViewer” metodes: <ul style="list-style-type: none"> • 2.2.3.1.1 renderer; • 2.2.3.1.2 onAfterRendering. Kontrollera klases “Models” metodes: <ul style="list-style-type: none"> • 2.2.3.1.9 onExpandAllXml; • 2.2.3.1.10 onCollapseAllXml.
1.3.2.3 JSON modeļa attēlošana	Kontroles klases “JSONViewer” metodes: <ul style="list-style-type: none"> • 2.2.3.2.1 renderer; • 2.2.3.2.2 onAfterRendering. Kontrollera klases “Models” metodes:

	<ul style="list-style-type: none"> • 2.2.3.2.11 onExpandAllJson; • 2.2.3.2.12 onCollapseAllJson.
1.3.2.4 Pēdējo ielādēto modeļu saraksta attēlošana sānjoslā	<p>Kontrollera klases “Models” metodes:</p> <ul style="list-style-type: none"> • 2.2.3.3.3 onToggleLastLoaded; • 2.2.3.3.7 updateModel; • 2.2.3.3.8 getDate.
1.3.3.1 Modeļa atjaunināšana	<p>Kontrollera klases “Models” metodes:</p> <ul style="list-style-type: none"> • 2.2.3.3.4 onReload; • 2.2.3.3.7 updateModel; • 2.2.3.3.8 getDate. <p>Klases “ChromeExtensionDataProvider” metodes:</p> <ul style="list-style-type: none"> • 2.2.5.2.3 checkForSapui; • 2.2.5.2.2 getModelData; <p>(Prasības realizācijas grafisku attēlojumu skatīt pielikumā “4. Prasības “1.3.3.1 Modeļa atjaunināšana” realizācijas grafisks attēlojums”)</p>
1.3.3.2 Visu modeļu atjaunināšana	<p>Kontrollera klases “Models” metodes:</p> <ul style="list-style-type: none"> • 2.2.3.3.5 onReloadAll; • 2.2.3.3.6 loadAllModels; • 2.2.3.3.7 updateModel; • 2.2.3.3.8 getDate. <p>Klases “ChromeExtensionDataProvider” metodes:</p> <ul style="list-style-type: none"> • 2.2.5.2.3 checkForSapui; • 2.2.5.2.1 getModelNames; • 2.2.5.2.2 getModelData; <p>(Prasības realizācijas grafisku attēlojumu skatīt pielikumā “3. Prasības “1.3.3.2 Visu modeļu atjaunināšana” realizācijas grafisks attēlojums”)</p>
1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski	<p>Klases “Message Handler” metodes:</p> <ul style="list-style-type: none"> • 2.2.5.1.1 subscribe. <p>2.2.4.3 Satura skripti/Content.js</p> <p>2.2.4.4 Fona lapa/Background.js</p> <p>(Prasības realizācijas grafisku attēlojumu skatīt pielikumā</p>

	“5. Prasības “1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski realizācijas grafisks attēlojums”)
1.3.4.1 Savienojuma izveidošana starp paplašinājumu un inspicējamo logu	2.2.4.4 Fona lapa/Background.js
1.3.4.2 Savienojuma pārtraukšana starp paplašinājumu un inspicējamo logu.	Klases “Message Handler” metodes: <ul style="list-style-type: none"> • 2.2.5.1.2 unsubscribe. 2.2.4.4 Fona lapa/Background.js
1.4 Nefunkcionālās prasības	2.3 Nefunkcionālo prasību realizācija

2.2.2 “SAPUI5 lietotne” uzbūve

Tā kā daļa “SAPUI5 lietotne” ir veidota izmantojot MVC principu, tad tā sastāv no:

- Kontrollera “Models” (sīkāk aprakstīts pie “SAPUI5 lietotnes” klašu dekompozīcijas “2.2.3.3 Kontrollera klase “Models””);
- Skata “Models”, jeb modeļu skata (sīkāk aprakstīts pie lietotāja saskarnes apraksta “2.4.1 Modeļu skats”);
- Modeļa, kas piesaistīts skatam (modeļa struktūru skatīt pielikumā “2.Skatam piesaistītā modeļa uzbūve”);
- Papildus klasēm:
 - Kontroles klases “XMLviewer” (sīkāk aprakstīts pie “SAPUI5 lietotnes” klašu dekompozīcijas “2.2.3.1 Kontroles klase “XMLViewer””);
 - Kontroles klases “JSONViewer” (sīkāk aprakstīts pie “SAPUI5 lietotnes” klašu dekompozīcijas “2.2.3.2 Kontroles klase “JSONViewer””);
 - Klases “XMLparser” (sīkāk aprakstīts pie “SAPUI5 lietotnes” klašu dekompozīcijas “2.2.3.4 Klase “XMLparser””).

2.2.3 “SAPUI5 lietotne” klašu dekompozīcija

2.2.3.1 Kontroles klase “XMLViewer”

Šīs klases metodes ir atbildīgas par jaunas kontroles “XMLViewer” izveidošanu un kontrolei paredzētā noformējuma un funkcionalitātes nodrošināšanu.

tabula 2.2

Metodes identifikators	Apraksts
2.2.3.1.1 renderer	Nodrošina kontroles “XMLViewer” atveidošanu ar atbilstošu noformējumu, izmantojot kontrolei piesaistītos JSON formāta datus (dati tiek piesaistīti metodē “2.2.3.3.2 <i>modelListFactory</i> ”).
2.2.3.1.2 onAfterRendering	Pēc kontroles “XMLViewer” atveidošanas, pievieno iespēju izvērst vai sakļaut kontrolē attēlotā modeļa sadaļas.

2.2.3.2 Kontroles klase “JSONViewer”

Šīs klases metodes ir atbildīgas par jaunas kontroles “JSONViewer” izveidošanu un kontrolei paredzētā noformējuma un funkcionalitātes nodrošināšanu.

tabula 2.3

Metodes identifikators	Apraksts
2.2.3.2.1 renderer	Nodrošina kontroles “JSONViewer” atveidošanu.
2.2.3.2.2 onAfterRendering	Izmantojot jQuery spraudni (avotu sk. “ <i>Izmantotā literatūra</i> ”), nodrošina kontrolei piesaistīto JSON formāta datu (dati tiek piesaistīti metodē “2.2.3.3.2 <i>modelListFactory</i> ”) attēlošanu ar formatējumu un iespēju izvērst vai sakļaut kontrolē attēlotā modeļa sadaļas.

2.2.3.3 Kontrolera klase “Models”

Šīs klases metodes ir atbildīgas par “Models” skata datu apstrādi.

tabula 2.4

Metodes identifikators	Apraksts
2.2.3.3.1 onInit	Tiek izsaukta pie skata inicializācijas, kad visas kontroles jau ir sagatavotas. Modificē skatu pirms tas ir attēlots un veic vienreizējas inicializācijas. Veic modeļu piesaistīšanu skatam un kontrolēm.
2.2.3.3.2 modelListFactory	Izmantojot skatam piesaistīto JSON modeli, nodrošina visu modeļu saraksta attēlošanu cilņu joslā. Kā arī nodrošina attiecīgā satura (fragmenta) attēlošanu cilnē

	<p>atkarībā no modeļa tipa (XML vai JSON), tas ir, nosaka vai cilnē attēlot fragmentu ar kontroli “XMLViewer” (“2.2.3.1 Kontroles klase “XMLViewer””) vai “JSONViewer”(“2.2.3.2 Kontroles klase “JSONViewer””). Attiecīgajam fragmentam piesaista arī konkrētā modeļa datus JSON formātā.</p>
2.2.3.3.3 onToggleLastLoaded	<p>Nodrošina iespēju lietotājam parādīt vai paslēpt sāņjoslu (nospiežot pogu “Show last loaded” vai “Hide last loaded”), kurā tiek attēloti pēdējo 15 ielādēto modeļu dati – modeļa nosaukums un laiks.</p> <p>Pēc pogas nospiešanas tiek izmainīti dati skatam piesaistītajā modelī, tādējādi tiek kontrolēta sāņjoslas redzamība un pogas teksts ar ikonu.</p>
2.2.3.3.4 onReload	<p>Nodrošina iespēju lietotājam pārlādēt kādu modeli manuāli (pēc pogas “Reload” nospiešanas).</p> <ol style="list-style-type: none"> 1. Tiek izsaukta metode “2.2.5.2.3 <i>checkForSapui</i>”, lai pārbaudītu vai ir pieejama SAPUI5 rīkkopa. 2. Ja ir pieejama, izsauc metodi “2.2.5.2.2 <i>getModelData</i>” no datu iegūšanas daļas “Chrome paplašinājums”, tai kā parametrus padodot simbolu virknes - modeļa nosaukumu un tipu -, ko iegūst no konteksta (piesaistītajiem datiem), kā arī atzvanīšanas metodi – “2.2.3.3.7 <i>updateModel</i>”. 3. Ja SAPUI5 rīkkopa nav pieejama, tiek izdots kļūdu paziņojums.
2.2.3.3.5 onReloadAll	<p>Nodrošina iespēju lietotājam pārlādēt visus modeļus manuāli (pēc pogas “Reload all models” nospiešanas).</p> <ol style="list-style-type: none"> 1. Tiek izsaukta metode “2.2.5.2.3 <i>checkForSapui</i>”, lai pārbaudītu vai ir pieejama SAPUI5 rīkkopa. 2. Ja ir pieejama, izsauc metodi “2.2.3.3.6 <i>loadAllModels</i>”. 3. Ja SAPUI5 rīkkopa nav pieejama, tiek izdots kļūdu paziņojums.
2.2.3.3.6 loadAllModels	<p>Šī metode nodrošina metodes “2.2.5.2.1</p>

	<p><i>getModelNames</i>” izsaukšanu no datu iegūšanas daļas “Chrome paplašinājums”, no kuras tiek iegūts masīvs ar visu modeļu nosaukumiem un tipiem. Pēc tam katram masīva elementam – modelim – tiek izsaukta metode “2.2.5.2.2 <i>getModelData</i>”, kurai tiek padots modeļa nosaukums, modeļa tips un atzvanīšanas funkcija - “2.2.3.3.7 <i>updateModel</i>”.</p>
2.2.3.3.7 <i>updateModel</i>	<p>Metodei tiek padotas simbolu virknes – modeļa nosaukums, tips un modeļa dati.</p> <ol style="list-style-type: none"> 1. Vispirms tiek parsēti modeļa dati JSON formātā, ja modeļa tips ir XML, tad izmantojot “XMLparser” klases metodi “2.2.3.4.1 <i>parse</i>”, ja modeļa tips JSON, tad izmantojot metodi <i>JSON.parse()</i>. Ja parsēšana ir bijusi veiksmīga, skatam piesaistītajā modelī atjauno modeļa datus un ielādēšanas laiku (ko iegūst veicot metodes “2.2.3.3.8 <i>getDate</i>” izsaukumu). 2. Ja parsēšana ir bijusi neveiksmīga, tiek izdots kļūdu paziņojums. 3. Tiek atjaunots arī sānjoslā esošo pēdējo 15 ielādēto modeļu saraksts – augšgalā tiek pievienots šī modeļa nosaukums un atjaunošanas laiks. Ja sarakstā ir vairāk par 15 ierakstiem, pēdējie tiek dzēsti.
2.2.3.3.8 <i>getDate</i>	Atgriež simbolu virkni ar pašreizējo datumu un laiku.
2.2.3.1.9 <i>onExpandAllXml</i>	Nodrošina kontrolē “XMLViewer” (“2.2.3.1 <i>Kontroles klase “XMLViewer”</i> ”) attēlotā modeļa visu sadaļu izvēršanu, lietotājam nospiežot pogu “Expand all”.
2.2.3.1.10 <i>onCollapseAllXml</i>	Nodrošina kontrolē “XMLViewer” (“2.2.3.1 <i>Kontroles klase “XMLViewer”</i> ”) attēlotā modeļa visu sadaļu sakļaušanu, lietotājam nospiežot pogu “Collapse all”.
2.2.3.2.11 <i>onExpandAllJson</i>	Nodrošina kontrolē “JSONViewer” (“2.2.3.2 <i>Kontroles klase “JSONViewer”</i> ”) attēlotā modeļa visu sadaļu izvēršanu, lietotājam nospiežot pogu “Expand all”.

2.2.3.2.12 onCollapseAllJson	Nodrošina kontrolē “JSONViewer” (“2.2.3.2 <i>Kontroles klase “JSONViewer”</i> ”) attēlotā modeļa visu sadaļu sakļaušanu, lietotājam nospiežot pogu “Collapse all”.
------------------------------	--

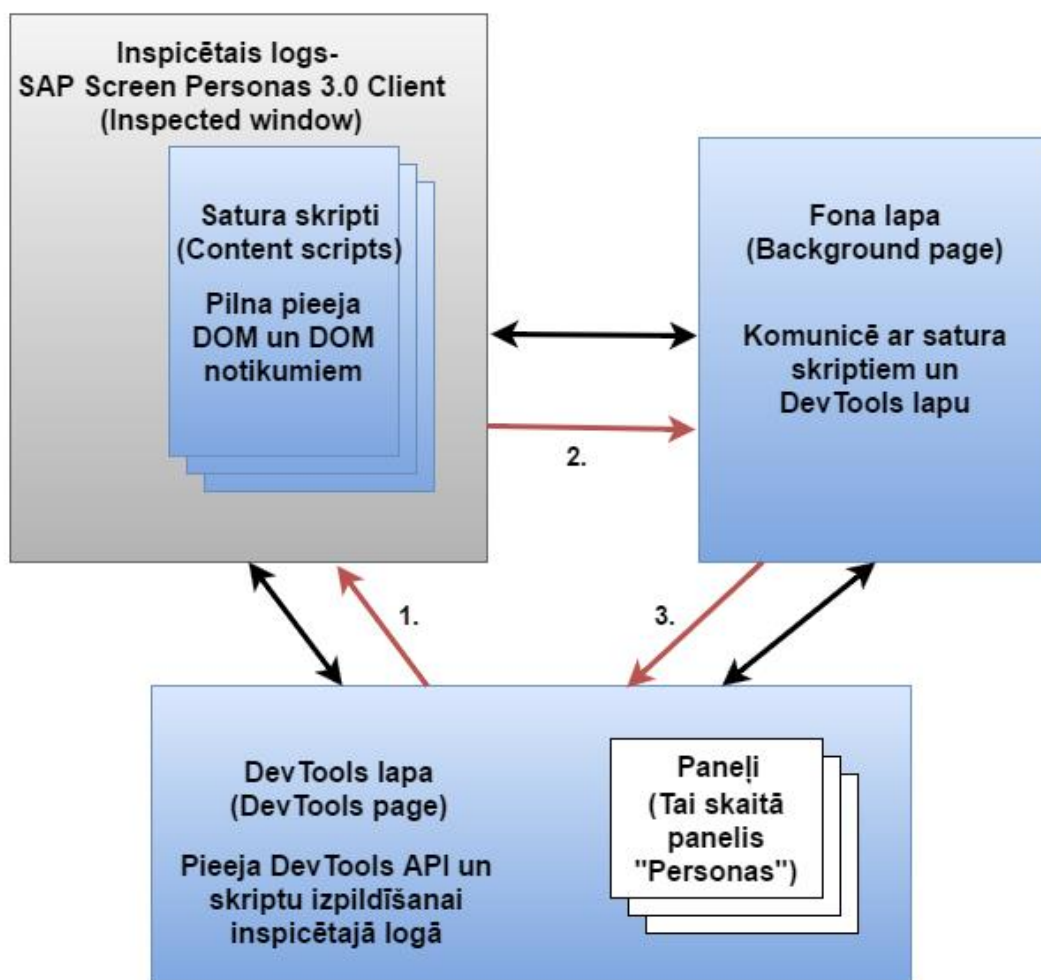
2.2.3.4 Klase “XMLparser”

Šīs klases metodes ir atbildīgas par XML simbolu virknes parsēšanu JSON formātā.

tabula 2.5

Metodes identifikators	Apraksts
2.2.3.4.1 parse	Rekursīvā metode nodrošina iespēju parsēt XML simbolu virkni JSON formātā. (piemēru ar ievadi un izvadi skatīt pielikumā “1. <i>Metodes “2.2.3.4.1 parse” ievaddatu un izvaddatu piemērs</i> ”)

2.2.4 “Chrome paplašinājums” uzbūve



att. 2.1 Chrome paplašinājuma struktūra

Izstrādātāja rīku paplašinājums pievieno papildus funkcionalitāti Chrome izstrādātāja rīkiem, tas ļauj pievienot jaunus lietotāja saskarnes paneļus un sānjoslas, kā arī komunicēt ar inspicējamo logu.

Izstrādātāja rīku paplašinājums ir strukturēts kā parasts paplašinājums. Tas sastāv no:

- fona lapas (“2.2.4.4 Fona lapa/Background.js”);
- satura skriptiem (“2.2.4.3 Satura skripti/Content.js”);
- citiem vienumiem (šajā projektā tās ir papildus klases, kuras sīkāk aprakstītas nodaļā “2.2.5 “Chrome paplašinājums” klašu dekompozīcija”);
- izstrādātāja rīku paplašinājums vēl papildus satur DevTools lapu, kurai ir pieeja DevTools API (“2.2.4.1 DevTools lapa/ DevTools.html”).

2.2.4.1 DevTools lapa/ DevTools.html

DevTools lapa – Katru reizi kad tiek atvērts izstrādātāja rīku logs, tiek izveidota instance paplašinājuma DevTools lapai, kura eksistē tik ilgi, kamēr ir atvērts izstrādātāja rīku logs. Šajā darbā tiek izmantotas šādas DevTools lapas iespējas:

- Izveidot jaunus paneļus un komunicēt ar tiem izmantojot devtools.panels API;
- Iegūt informāciju par inspicējamo logu un izpildīt tajā skriptu izmantojot devtools.inspectedWindow API.

2.2.4.2 Inspicējamais logs/SAP Screen Personas 3.0 Client

Pārlūkprogrammas Google Chrome cilnē atvērtais logs, ar kuru notiek komunikācija un no kuras tiek iegūti dati.

2.2.4.3 Satura skripti/Content.js

JavaScript kods, kas tiek izpildīts tīmekļa lapas kontekstā. Satura skripti tiek izpildīti izolētā vidē, līdz ar to tiem ir pieeja lapas DOM objektam, bet ne JavaScript funkcijām un mainīgajiem, ko ir izveidojusi lapa. Šajā darbā tiek izmantotas šādas satura skriptu iespējas:

- Ziņu saņemšana no inspicētā loga;
- Ziņu sūtīšana uz paplašinājuma fona lapu.

2.2.4.4 Fona lapa/Background.js

Fona lapa nodrošina komunikāciju starp satura skriptiem un DevTools lapu. Šajā darbā tiek izmantotas šādas notikumu lapas iespējas:

- Ziņas saņemšana no satura skriptiem;
- Ziņas nosūtīšanu tālāk uz DevTools lapu;
- Savienojuma izveidošana starp inspicējamo logu un paplašinājumu (tas tiek nodrošināts izveidojot savienojumu starp fona lapu un paplašinājuma DevTools lapu).
- Savienojuma pārtraukšana starp inspicējamo logu un paplašinājuma DevTools lapu.

2.2.4.5 Izvēlētais ceļš saziņai starp inspicējamo logu un paplašinājumu

Attēlā “att. 2.1 **Chrome paplašinājuma struktūra**” ar melnajām bultiņām ir atzīmēti iespējamie komunikācijas virzieni, taču ar sarkanajām ir atzīmēti paplašinājuma izstrādē izmantotie. Ejot pa katru bultiņu tiek veiktas noteiktas darbības:

1. Ar devtools.inspectedWindow API palīdzību inspicētajā logā tiek izpildīts skripts, kas pievieno notikumu uztvērējus lapas kodolā esošajiem modeļiem, lai sekotu līdz to izmaiņām.

2. Notikumu uztvērējam pamanot izmaiņas kādā modelī, tiek nosūtīta ziņa uz satura skriptiem. No satura skriptiem tā tiek tālāk nosūtīta uz fona lapu.
3. Fona lapā tiek saņemta ziņa un nosūtīta tālāk uz DevTools lapu, kur tā tiek apstrādāta.

2.2.5 "Chrome paplašinājums" klašu dekompozīcija

2.2.5.1 Klase "Message Handler"

Šīs klases funkcijas ir atbildīgas par ziņu apmaiņu starp paplašinājumu un inspicējamo logu, kā arī saņemto ziņu apstrādi.

tabula 2.6

Metodes identifikators	Apraksts
2.2.5.1.1 subscribe	<p>Metodei tiek padota simbolu virkne – modeļa nosaukums un atzvanīšanas funkcija.</p> <p>Nodrošina iespēju pieteikties jaunumu saņemšanai kādam noteiktam modelim – saglabājot modeļa nosaukumu un atzvanīšanas funkciju masīvā. Kad tiek saņemta ziņa paplašinājumā, nodrošina tālākās funkcijas izsaukšanu atbilstoši ziņas nosaukumam – pārbauda vai masīvā ir atslēga ar ziņas nosaukumu, ja ir tad izsauc atslēgai pakārtoto funkciju.</p>
2.2.5.1.2 unsubscribe	<p>Metodei tiek padots modeļa nosaukums – simbolu virkne.</p> <p>Nodrošina iespēju atteikties no jaunumu saņemšanas kādam noteiktam modelim – dzēšot modeļa nosaukumu no masīva.</p>

2.2.5.2 Klase "Chrome Extension Data Provider"

Šīs klases metodes nodrošina nepieciešamo funkcionalitāti datu iegūšanai no inspicētā loga.

tabula 2.7

Metodes identifikators	Apraksts
2.2.5.2.1 getModelNames	<p>Nodrošina iespēju iegūt visu modeļu nosaukumus un tipus (XML vai JSON), kas ir reģistrēti inspicējamā loga kodolā (SAP Screen Personas 3.0 Client) izmantojot chrome.devtools.inspectedWindow API. Atgriež masīvu ar</p>

	modeļu nosaukumiem un tipiem.
2.2.5.2.2 getModelData	<p>Metodei tiek padotas simbolu virknes – modeļa nosaukums un tips, kā arī atzvanīšanas funkcija – “2.2.3.3.7 updateModel”.</p> <p>Nodrošina iespēju iegūt datus no XML vai JSON modeļa, kas ir reģistrēts inspicējamās lapas kodolā, izmantojot chrome.devtools.inspectedWindow API.</p> <p>Ja modelim nav uzstādīti dati, tiek izdots kļūdu paziņojums.</p> <p>Izsauc atzvanīšanas funkciju “2.2.3.3.7 updateModel”, kurai padod simbolu virknes – modeļa datus, nosaukumu un tipu.</p>
2.2.5.2.3 checkForSapui	Nodrošina iespēju pārbaudīt vai inspicētajā logā ir pieejama SAPUI5 rīkkopa, izmantojot chrome.devtools.inspectedWindow API.

2.3 Nefunkcionālo prasību realizācija

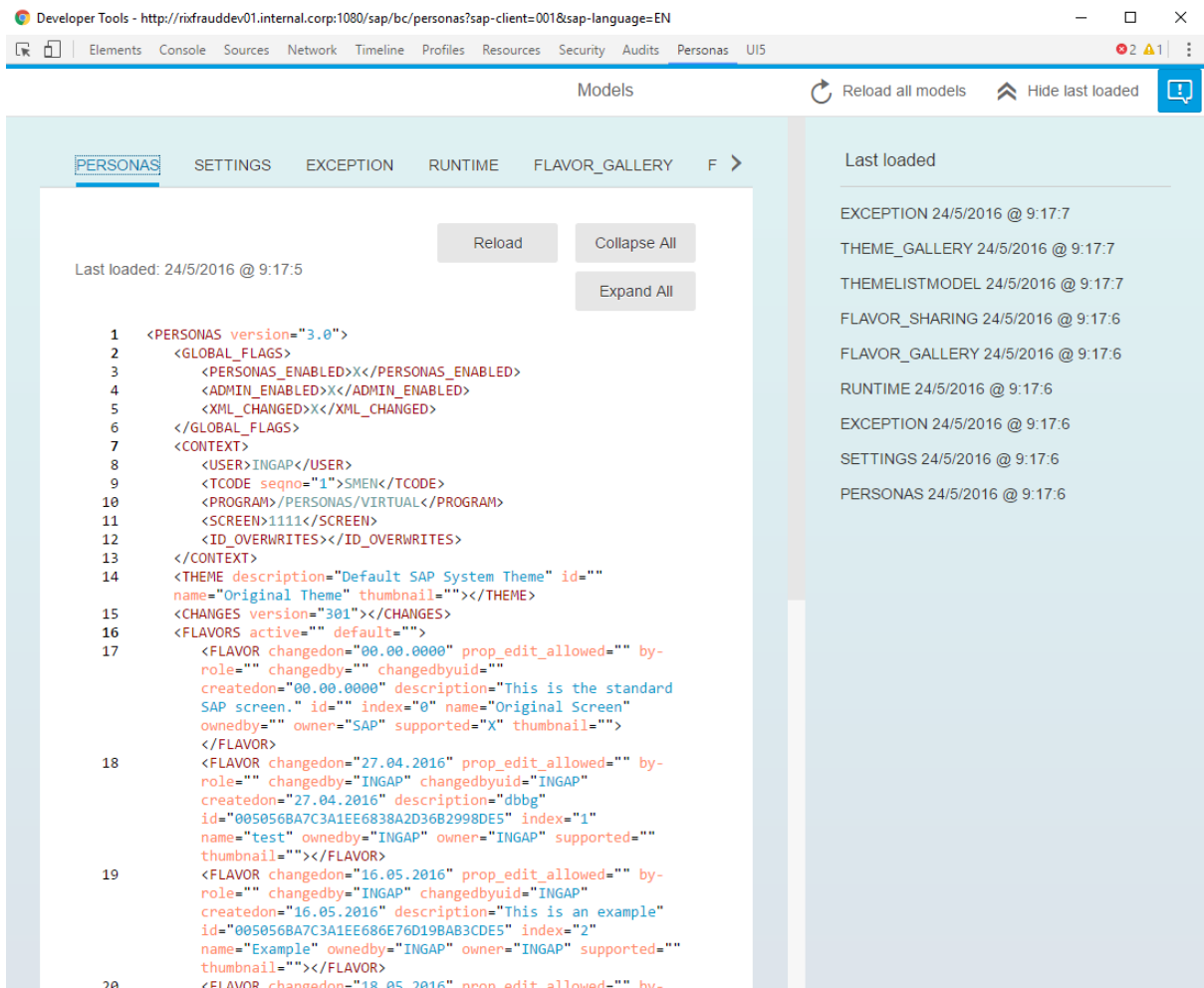
tabula 2.8

Nefunkcionālā prasība	Realizācija
1.4.1.1 JSDoc	Programmatūras izstrādes laikā pirmkods tika komentēts ar JSDoc komentāriem.
1.4.1.2 Lietotāja dokumentācija	Atbilstoši izstrādātajam paplašinājumam tika izstrādāta arī lietotāja dokumentācija angļu valodā.
1.4.2.1 Problēmu un uzdevumu pārvaldība	Programmatūras uzdevumu un problēmu pieteikumu pārvaldībai tika izmantota Atlassian JIRA.
1.4.3.1 Instalēšana un atinstalēšana	Lietotāja dokumentācijā pa soļiem tika aprakstīta paplašinājuma instalēšana un atinstalēšana.

2.4 Lietotāja saskarnes apraksts

Šajā sadaļā tiek apskatīts SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma ekrānformas.

2.4.1 Modeļu skats

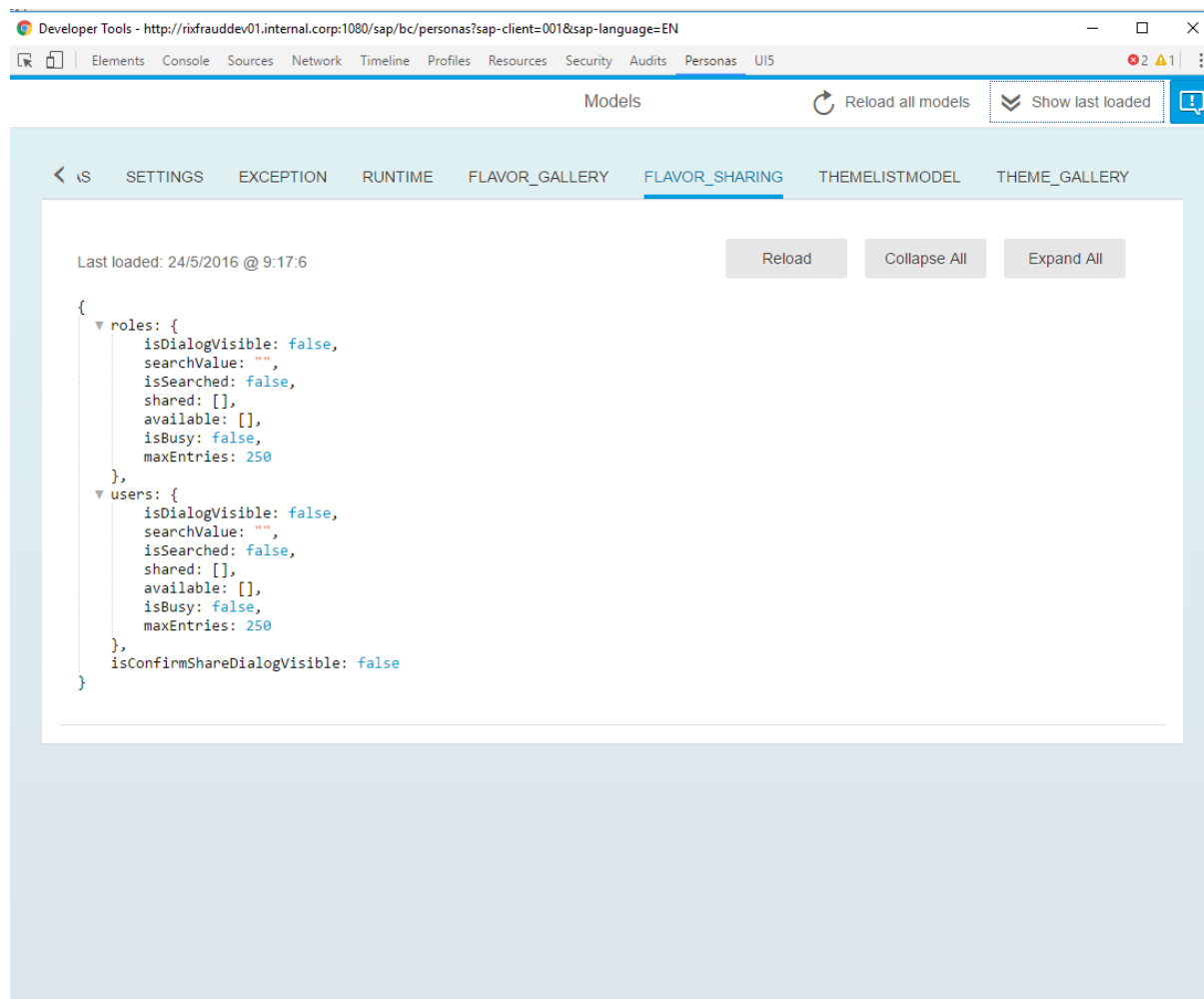


att. 2.2 Lietotāja saskarne – modeļu skats ar izvēlētu XML modeli

Attēlā ir redzams Google Chrome izstrādātāja rīku logs, kurā ir atvērts panelis “Personas”. Lapas galvenes labajā pusē ir redzama poga “Reload all models”, kuru nospiežot tiek pārlādēti visi modeļi un poga “Hide last loaded”, ar kuru iespējams paslēpt vai parādīt labajā pusē esošo sāņjoslu “Last loaded”, kurā tiek attēloti pēdējo 15 atjaunoto modeļu nosaukumi un pārlādēšanas laiks, kā arī paziņojumu poga, uz kuras nospiežot parādās uznirstošā izvēlne ar paziņojumiem (par paziņojumu attēlošanu sīkāk nākamajā daļā - “2.4.2 Paziņojumu parādīšana”).

Lapas galvenajā daļā ir redzama cilņu josla, kur katru cilni veido XML vai JSON skats, cilnes nosaukumā ir attiecīgā modeļa nosaukums. Šajā attēlā ir redzams XML skats, kurā ir attēlots modelis “PERSONAS”. Gan XML, gan JSON skatos atrodas poga “Reload”, kuru nospiežot tiek atjaunināts aktīvais modelis, poga “Collapse all”, kuru nospiežot tiek sakļautas

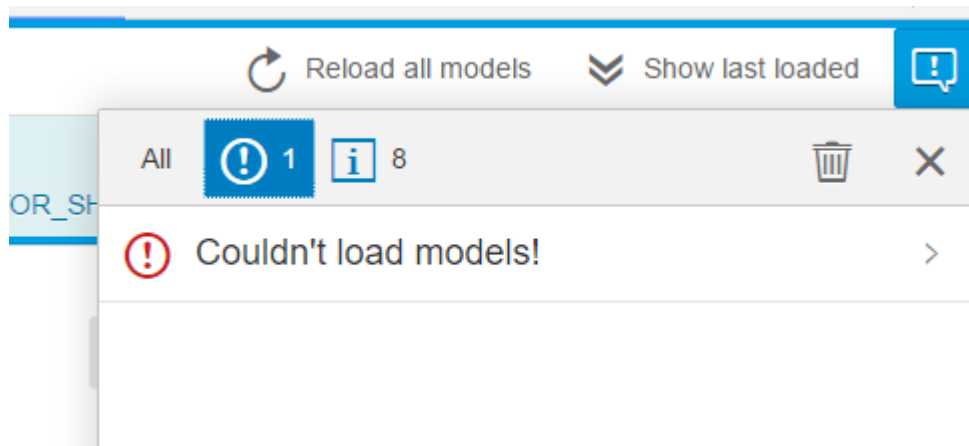
visas sadaļas modelī un poga “Expand all”, kuru nospiežot tiek izvērstas visas sadaļas modelī.



att. 2.3 Lietotāja saskarne – modeļu skats ar atvērtu JSON modeli

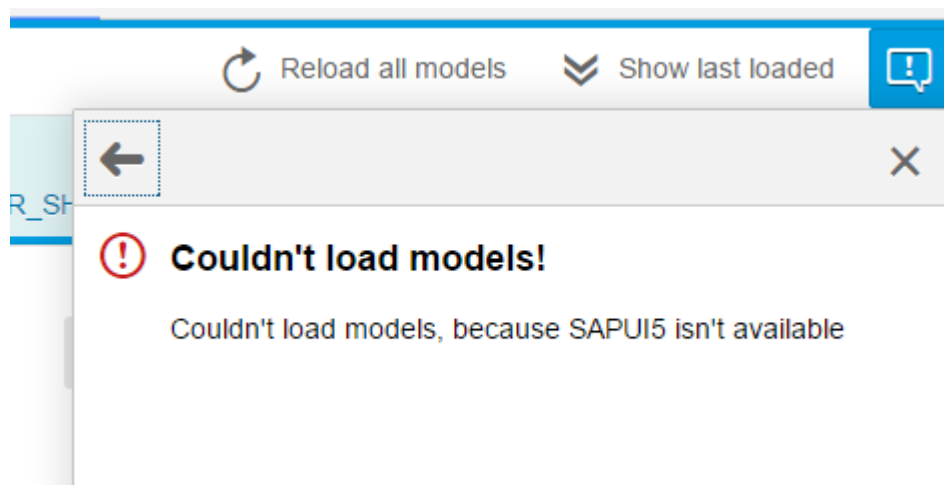
Šajā attēlā arī ir redzams modeļu skats, taču atšķirībā no attēla “att. 2.2 Lietotāja saskarne – modeļu skats ar izvērtu XML modeli”, sāņjosla “Last loaded” ir paslēpta un cilņu joslā ir izvēlēta cilne, kurā ir redzams JSON modelis “FLAVOR_SHARING”.

2.4.2 Paziņojumu parādīšana



att. 2.4 Lietotāja saskarne – paziņojumu attēlošana

Paziņojumi tiek parādīti uznirstošajā izvēlnē pēc paziņojumu pogas nospiešanas. Paziņojumu poga ir izcelta tikai tad, ja ir kāds nolasīts paziņojums. Tiek rādīti 4 veidu paziņojumi – kļūdas, brīdinājuma, veiksmes un informācijas -, uznirstošajā izvēlē var atlasīt, kurus attēlot – visus vai kāda konkrēta veida paziņojumus. Uznirstošajā izvēlnē ir arī poga paredzēta visu paziņojumu dzēšanai.



att. 2.5 Lietotāja saskarne – paziņojuma apraksta attēlošana

Nospiežot uz paziņojuma tiek atvērts paziņojuma apraksts un pēc izlasīšanas tas tiek dzēsts no paziņojumu saraksta.

3. TESTĒŠANAS DOKUMENTĀCIJA

3.1 Ievads

Testēšanas dokumentācija sastāv no 3 nodaļām:

1. Ievads – nodaļā ir īsi aprakstīts šī dokumenta nolūks un testēšanas metodika
2. Trasējamības tabula - attēlots kā visas PPS izvirzītās prasības ir testētas – tiek norādīti attiecīgo testēšanas scenāriju identifikatori katrai prasībai;
3. Testēšanas žurnāls– aprakstīts katrs testēšanas scenārijs, sagaidāmie rezultāti un testēšanas rezultāti;
4. Testēšanas kļūdu apstrāde – nodaļā aprakstītas testēšanas kļūdas un to novēršana, kā arī kļūdas atkārtotas testēšanas rezultāts.

3.1.1 Nolūks

Šī testēšanas dokumentācija ir paredzēta SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma veikto testu aprakstīšanai. Pēc aprakstītās metodikas, testēšanu veica gan izstrādātājs, gan pasūtītājs, lai apstiprinātu atbilstību pret PPS izvirzītajām prasībām.

3.1.2 Definīcijas

Skatīt “*VĀRDNĪCA*” (8.lpp.).

3.1.3 Saistība ar citiem dokumentiem

Testēšanas dokumentācija ir saistīta ar SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma programmatūras prasību specifikāciju.

3.1.4 Testēšanas metodika

Testēšana tika veikta izmantojot melnās kastes metodi - tika testēta tieši izstrādātās programmatūras uzvedība. Izvēlējās šo testēšanas principu, jo tas fokusējas uz programmatūras funkcionālajām prasībām – testēšanas laikā tika izvēlēti tādi testpiemēri, kas pārbauda visas PPS izvirzītās funkcionālās prasības. Testēšanas atbilstību pret izvirzītajām prasībām skatīt “*3.2 Trasējamības tabula*”. Testēšanas žurnālā apkopotie testu iznākumi attiecināmi uz sistēmas testēšanu pēc pēdējās izstrādes iterācijas pabeigšanas

Testēšana tika veikta manuāli pildot testu scenārijus. Tā kā izstrādātā programmatūra ir paplašinājums Google Chrome izstrādātāja rīkiem, netika atrasts neviens šīs

pārlūkprogrammas automatizācijas rīks, kas ierakstītu darbības, kuras tiek veiktas izstrādātāja rīku logā (tika apskatīts: Selenium IDE spraudnis pārlūkprogrammai – darbojas tikai uz Firefox, Chromium browser automation, iMacros – neieraksta darbības, kas veiktas izstrādātāja rīkos).

Testēšanas scenārijos, kur bija nepieciešams mainīt modeļa datus, netika izmantoti SAP Screen Personas 3.0 Client kodolā reģistrētie modeļi, bet gan tika reģistrēti jauni - eksperimentāli. SAP Screen Personas 3.0 modeļu izmaiņšana var izraisīt programmatūras atteici.

Testēšanas laikā tika pārbaudīta arī HTML un CSS datņu atbilstība HTML5 un CSS3 standartiem. Kā arī tika pārbaudīts nefunkcionālo prasību izpildījums.

3.2 Trasējamības tabula

tabula 3.1

PPS prasības identifikators	Veiktā testa scenārija identifikators
1.3.2.1 Visu modeļu saraksta attēlošana	3.3.1 Testa Nr. 1 scenārijs
1.3.2.2 XML modeļa attēlošana	3.3.2 Testa Nr. 2 scenārijs
1.3.2.3 JSON modeļa attēlošana	3.3.3 Testa Nr. 3 scenārijs
1.3.2.4 Pēdējo ielādēto modeļu saraksta attēlošana sānjoslā	3.3.4 Testa Nr. 4 scenārijs, 3.3.5 Testa Nr. 5 scenārijs, 3.3.6 Testa Nr. 6 scenārijs
1.3.3.1 Modeļa atjaunināšana	3.3.4 Testa Nr. 4 scenārijs
1.3.3.2 Visu modeļu atjaunināšana	3.3.5 Testa Nr. 5 scenārijs
1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski	3.3.6 Testa Nr. 6 scenārijs
1.3.4.1 Savienojuma izveidošana starp paplašinājumu un inspicējamo logu	3.3.6 Testa Nr. 6 scenārijs
1.3.4.2 Savienojuma pārtraukšana starp paplašinājumu un inspicējamo logu.	3.3.6 Testa Nr. 6 scenārijs
1.4 Nefunkcionālās prasības	3.3.7 Citi testi

3.3 Testēšanas žurnāls

3.3.1 Testa Nr. 1 scenārijs

Šī testa scenārija mērķis ir pārliccināties, ka skatam tiek piesaistīts korekts modelis, ka modeļa dati ir piesaistīti atbilstošajām kontrolēm un izņēmumu gadījumos – ja modelis ir tukšs – nerodas kļūdas.

tabula 3.2

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts/Kļūda
1.	Atver Chrome DevTools logu un izvēlas paneli “Personas”, skatam piesaistītajā modelī nav datu.	Skatā tiek parādīts “No data”.	24.05.2016 - OK
2.	Atver Chrome DevTools logu un izvēlas paneli “Personas”, skatam piesaistītajā modelī ir dati.	Skatā tiek attēlots saraksts ar modeļiem cilņu veidā.	24.05.2016 - OK

3.3.2 Testa Nr. 2 scenārijs

Šī testa scenārija mērķis ir pārliccināties, ka XML modelis tiek attēlots atbilstoši PPS izvirzītajām prasībām pret noformējumu un ir nodrošinātas arī tādas iespējas kā sadaļu sakļaušana un izvēršana, visu sadaļu sakļaušana un visu sadaļu izvēršana.

tabula 3.3

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts
3.	Izvēlas cilni ar XML modeli.	Izvēlētā modeļa dati tiek attēloti ar PPS izvirzītajā prasībā “1.3.2.2 XML modeļa attēlošana” noteikto formatējumā.	24.05.2016 - OK
4.	Nospiež uz pogas “Expand all”	Attēlotajā XML modelī tiek izvērstas visas sadaļas.	24.05.2016 - OK
5.	Nospiež uz pogas “Collapse all”	Attēlotajā XML modelī tiek sakļautas visas sadaļas.	24.05.2016 - OK

3.3.3 Testa Nr. 3 scenārijs

Šī testa scenārija mērķis ir pārliccināties, ka JSON modelis tiek attēlots atbilstoši PPS izvirzītajām prasībām pret noformējumu un ir nodrošinātas arī tādas iespējas kā sadaļu sakļaušana un izvēršana, visu sadaļu sakļaušana un visu sadaļu izvēršana.

tabula 3.4

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts
6.	Izvēlas cilni ar JSON modeli.	Izvēlētajā modeļa dati tiek attēloti ar PPS izvirzītajā prasībā “1.3.2.3 JSON modeļa attēlošana” noteikto formatējumu.	24.05.2016 - OK
7.	Nospiež uz pogas “Expand all”	Attēlotajā JSON modelī tiek izvērstas visas sadaļas.	24.05.2016 - OK
8.	Nospiež uz pogas “Collapse all”	Attēlotajā JSON modelī tiek sakļautas visas sadaļas.	24.05.2016 - OK

3.3.4 Testa Nr. 4 scenārijs

Šī testa scenārija mērķis ir pārliccināties, vai modeļu atjaunošana manuāli (ar pogas “Reload” nospiešanu) izpildās atbilstoši PPS izvirzītajām prasībām un kļūdu gadījumos, tās tiek pareizi apstrādātas un izdoti atbilstošie kļūdu paziņojumi.

tabula 3.5

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts
9.	Nospiež pogu “Reload” modelim <modeļa nosaukums>, kad SAPUI5 rīkkopa nav pieejama.	Dati netiek atjaunoti, pie kļūdu paziņojumiem parādās “Couldn’t load model <modeļa nosaukums>, because SAPUI5 isn’t available!”. Arī pēdējo ielādēto modeļu sarakstā sānjoslā netiek pievienots modeļa nosaukums.	24.05.2016 - OK
<p>Nākamajos soļos ir jāveic izmaiņas modeļa datos, tāpēc, lai neizraisītu SAP Screen Personas 3.0 Client atteici mainot jau esošos modeļus, tiek izveidoti jauni eksperimentāli XML un JSON modeļi, kā arī tiem tiek uzstādīti dati, DevTools konsolē izpildot skriptu:</p> <pre>var oXMLTestModel = new frames[0].sap.ui.model.xml.XMLModel(); frames[0].sap.ui.getCore().setModel(oXMLTestModel, "XMLTestModel"); oXMLTestModel.setXML("<XMLTestModel>This is a test model</XMLTestModel>");</pre>			

<pre> var oJSONTestModel = new frames[0].sap.ui.model.json.JSONModel(); frames[0].sap.ui.getCore().setModel(oJSONTestModel, "JSONTestModel"); oJSONTestModel.setJSON("{\"test\":true}"); </pre> <p>Lai modeļi parādītos cilņu joslā ir jānospiež poga “Reload all models”.</p> <p>Lai šo prasību varētu notestēt, vispirms ir jānoņem notikumu uztvērēji, kas pievienoti tikko izveidotajiem modelim, lai izmaiņu gadījumā modelis netiktu automātiski atjaunināts pirms pogas “Reload” nospiešanas. Caur DevTools loga konsoli inspicējamā logā izpilda skriptu:</p> <pre> frames[0].bindings.JSONTestModel.destroy(); frames[0].bindings.XMLTestModel.destroy(); </pre>			
10.	<p>Caur Google Chrome konsoli inspicētajā logā izpilda skriptu, kas tikko izveidotajam JSON modelim “JSONTestModel” uzstāda kļūdainus datus:</p> <pre>frames[0].sap.ui.getCore().getModel("JSONTestModel").setJSON("{\"test\":true}");</pre> <p>Nospiež pogu “Reload” un modeļa dati inspicētajā loga nav pareizi</p>	<p>Dati netiek atjaunoti, pie kļūdu paziņojumiem parādās “Couldn’t load model JSONTestModel, because there is an error in JSON format!”.</p> <p>Arī pēdējo ielādēto modeļu sarakstā sānjoslā netiek pievienots modeļa nosaukums.</p>	<p>24.05.2016 - Kļūda (skatīt “3.4 Testēšanas kļūdu apstrāde”)</p> <p>25.05.2016 - OK</p>
11.	<p>Caur Google Chrome konsoli inspicētajā logā izpilda skriptu, kas tikko izveidotajam XML modelim “XMLTestModel” uzstāda kļūdainus datus:</p> <pre>frames[0].sap.ui.getCore().getModel("XMLTestModel").setXML("<XMLTestModel>");</pre> <p>Nospiež pogu “Reload” un modeļa dati inspicētajā loga nav pareizi</p>	<p>Dati netiek atjaunoti, pie kļūdu paziņojumiem parādās “Couldn’t load model XMLTestModel, because there is an error in XML format!”.</p> <p>Arī pēdējo ielādēto modeļu sarakstā sānjoslā netiek pievienots modeļa nosaukums.</p>	<p>24.05.2016 - OK</p>
12.	<p>“JSONTestModel” uzstāda korektus datus (tos ir iespējams noparsēt):</p> <pre>frames[0].sap.ui.getCore().getModel("JSONTestModel").setJSON("{\"test\":false}");</pre> <p>Nospiež pogu “Reload”.</p>	<p>Tiek atjaunoti modeļa dati un pēdējais pārlādēšanas laiks.</p> <p>Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienots modeļa nosaukums.</p>	<p>24.05.2016 - OK</p>
13.	<p>“XMLTestModel” uzstāda korektus</p>	<p>Tiek atjaunoti modeļa dati un</p>	<p>24.05.2016 -</p>

	<p>datus (tos ir iespējams noparsēt):</p> <pre>frames[0].sap.ui.getCore().getModel("XMLTestModel").setXML("<XMLTestModel>This is a test model</XMLTestModel>");</pre> <p>Nospiež pogu "Reload".</p>	<p>pēdējais pārlādēšanas laiks.</p> <p>Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienots modeļa nosaukums</p>	OK
--	---	--	----

3.3.5 Testa Nr. 5 scenārijs

Šī testa scenārija mērķis ir pārliecināties, vai visu modeļu atjaunošana (ar pogas "Reload all models" nospiešanu) izpildās atbilstošo PPS izvirzītajām prasībām un kļūdu gadījumos, tās tiek pareizi apstrādātas un izdoti atbilstošie kļūdu paziņojumi.

tabula 3.6

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts
14.	Nospiež pogu "Reload all models", kad SAPUI5 nav pieejams.	Dati netiek atjaunoti, pie kļūdu paziņojumiem parādās "Couldn't load models, because SAPUI5 isn't available!". Arī pēdējo ielādēto modeļu sarakstā sānjoslā netiek pievienoti modeļu nosaukumi.	24.05.2016 - OK
<p>Nākamajos soļos ir jāveic izmaiņas modeļa datos, tāpēc, lai neizraisītu SAP Screen Personas 3.0 programmatūras atteici, tiek izveidoti jauni XML un JSON modeļi, bet tiem netiek uzstādīti dati:</p> <pre>var oXMLTestModel = new frames[0].sap.ui.model.xml.XMLModel(); frames[0].sap.ui.getCore().setModel(oXMLTestModel, "XMLTestModel"); var oJSONTestModel = new frames[0].sap.ui.model.json.JSONModel(); frames[0].sap.ui.getCore().setModel(oJSONTestModel, "JSONTestModel");</pre>			
15.	Nospiež pogu "Reload all models" un tikko izveidotajam modelim "JSONTestModel" nav uzstādīti dati.	Tiek atjaunoti dati visiem modeļiem izņemot "JSONTestModel", pie kļūdu paziņojumiem parādās "Couldn't load model JSONTestModel, because there is no data set to this model!". Pēdējo ielādēto modeļu sarakstā	24.05.2016 - Kļūda (skatīt "3.4 Testēšanas kļūdu apstrāde") 25.05.2016 - OK

		sānjoslā tiek pievienoti visu modeļu nosaukumi, izņemot neatjaunināto modeli – “JSONTestModel”.	
16.	Nospiež pogu “Reload all models” un tikko izveidotajam modelim “XMLTestModel” nav uzstādīti dati.	Tiek atjaunoti dati visiem modeļiem izņemot “XMLTestModel”, pie kļūdu paziņojumiem parādās “Couldn’t load model XMLTestModel, because there is no data set to this model!”. Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienoti visu modeļu nosaukumi, izņemot neatjaunoto modeli – XMLTestModel”.	24.05.2016 - Kļūda (skatīt “3.4 Testēšanas kļūdu apstrāde”) 25.05.2016 - OK
17.	“XMLTestModel” uzstāda kļūdainus datus: oXMLTestModel.setXML("<XMLTestModel>"); Nospiež pogu “Reload all models” un modelim nav uzstādīti pareizi dati.	Tiek atjaunoti dati visiem modeļiem izņemot “XMLTestModel”, pie kļūdu paziņojumiem parādās “Couldn’t load model XMLTestModel, because there is an error in XML format!”. Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienoti visu modeļu nosaukumi, izņemot neatjaunoto modeli – “XMLTestModel”.	24.05.2016 - OK
18.	Nospiež pogu “ReloadAll”, SAPUI5 rīkkopa ir ielādējusies, visiem inspicējamā logā reģistrētajiem modeļiem ir uzstādīti dati un tie ir pareizi (tos ir iespējams noparsēt).	Tiek atjaunoti dati visiem modeļiem. Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienoti visu modeļu nosaukumi.	24.05.2016 - OK

3.3.6 Testa Nr. 6 scenārijs

Šī testa scenārija mērķis ir pāliecināties, ka izmaiņu gadījumā modeļa dati tiek atjaunoti automātiski – ir izveidots savienojums starp inspicējamo logu un paplašinājumu un pa šo savienojumu tiek sūtītas ziņas.

tabula 3.7

Nr.	Ievaddati	Sagaidāmais rezultāts	Rezultāts/Kļūda
19.	Caur Google Chrome konsoli inspicētajā logā (SAP Screen Personas 3.0 Client) izpilda skriptu, kas jau esošam modelim izmaina datus vai arī SAP Screen Personas 3.0 Client veic izmaiņas lietotāja saskarnē, lai modeļa dati izmainītos	Skatā tiek automātiski parādīti modeļa jaunākie dati un pēdējais modeļa atjaunināšanas laiks. Pēdējo ielādēto modeļu sarakstā sānjoslā tiek pievienots modeļa nosaukums.	24.05.2016 - OK

3.3.7 Citi testi

tabula 3.8

Nefunkcionālā prasība	Testēšanas pieraksti
1.4.1.1 JSDoc	No izstrādātā pirmkoda tika ģenerēta automātiskā dokumentācija (sk. “8. JSDoc”)
1.4.1.2 Lietotāja dokumentācija	Klientam tika piegādāta programmatūra kopā ar lietotāja dokumentāciju angļu valodā.
1.4.2.1 Problēmu un uzdevumu pārvaldība	Programmatūras uzdevumu un problēmu pieteikumu pārvaldībai tika izmantota Atlassian JIRA. Tika pārbaudīts vai ir atrisinātas visas problēmas un uzdevumi, kas nepieciešami, lai varētu klientam piegādāt programmatūru.
1.4.3.1 Instalēšana un atinstalēšana	Lietotāja dokumentācijā ir iekļauta nodaļa ar paplašinājuma instalēšanas un atinstalēšanas pamācību.

3.4 Testēšanas kļūdu apstrāde

tabula 3.9

Testa Nr.	Kļūda	Risinājums	Rezultāts
10.	Testēšanas laikā tika secināts, ka inspicējamā logā JSON modelim nav iespējams uzstādīt kļūdainus datus, jo pirms uzstādīšanas tie tiek parsēti. Tātad nav iespējama tāda situācija, ka ir reģistrēts JSON modelis ar kļūdainiem datiem.	Prasība pret kļūdu ziņojumu, kā arī tests tiek atcelts.	-
15.	Testēšanas laikā tika secināts, ka, ja JSON modelim nav uzstādīti dati, tad DevTools konsolē izpildot: <code>frames[0].sap.ui.getCore().getModel("JSONTestModel").getJSON();</code> tiek atgriezts "{}", nevis "undefined".	Jāizlabo pārbaude – ja atgriež "{}" (nevis "undefined"), tad dati nav uzstādīti.	25.05.2016 - OK
16.	Testēšanas laikā tika secināts, ka, ja XML modelim nav uzstādīti dati, tad DevTools konsolē izpildot <code>frames[0].sap.ui.getCore().getModel("XMLTestModel").getXML();</code> tiek izmesta kļūda, nevis atgriezts "undefined".	Jāveic šīs kļūdas apstrāde.	25.05.2016 - OK

4. PROJEKTA ORGANIZĀCIJA

Programmatūra tika izstrādāta pielietojot iteratīvo pieeju, jo sākumā galvenās prasības bija definētas aptuveni, taču pietiekami, lai varētu sākt darbu pie programmatūras izstrādes. Turpmākajās iterācijās visas prasības tika konkretizētas, katras iterācijas sākumā organizējot tikšanās ar programmatūras pasūtītāju un papildinot programmatūras prasību specifikāciju. Katras iterācijas beigās tika veikta pievienotās funkcionalitātes testēšana.

Izstrādātās sistēmas testēšana un prasību akceptēšana tika veikta gan no izstrādātāja puses, gan arī no pasūtītāja puses. No izstrādātāja puses sākumā paplašinājuma testēšana tika veikta izmantojot pagaidu eksperimentālu lapu, taču, projektam attīstoties, vēlāk tika testēts izmantojot SAP Screen Personas 3.0 Client, kam šī programmatūra ir paredzēta. Pasūtītājs veica programmatūras testēšanu un akceptēšanu, gan katras iterācijas beigās, gan izstrādes beigās izmantojot SAP Screen Personas 3.0 Client.

Projekta pārvaldībai tikai izmantota Atlassian JIRA (sk. “7. *Atlassian JIRA uzdevumu un problēmu pieteikumi*”), kur tika definēti veicamie uzdevumi un bija iespējams arī paredzēt uzdevuma izpildes laiku, kā arī atzīmēt uzdevuma reālo izpildes laiku. Lai projektam piesaistītos uzdevumus būtu vieglāk pārskatīt, tie tika strukturēti apakšuzdevumos.

5. KVALITĀTES NODROŠINĀŠANA

Lai izstrādātajam produktam nodrošinātu pēc iespējas augstāku kvalitāti, izstrādes procesā tika ievēroti vairāki noteikumi, kuri attiecas gan uz programmatūras koda izstrādi un noformēšanu, gan arī uz pašu izstrādes procesu.

Lai JavaScript koda izstrādes procesā rastos pēc iespējas mazāk kļūdas un kods atbilstu JavaScript stila vadlīnijām, izmantoju ESLint sparudni. JavaScript koda kvalitāte tika pārbaudīta arī izmantojot SonarQube (sk. “6. *SonarQube ekrānuņēmumi*”), kura atskaites sniedza informāciju par dublikāta koda rindiņām, koda sarežģītību, programmēšanas un noformēšanas kļūdām, kā arī informāciju par iespējamiem uzlabojumiem. Atskaites tika ņemtas vērā, dublikāta koda rindiņas tika samazinātas līdz 0.0% un atrastās problēmas un kļūdas tika izlabotas.

Kā arī sistēmas izstrādē, tika ievēroti W3C organizācijas izstrādātie HTML5 un CSS3 standarti. Kods tika validēts tiešsaistē sekojošās vietnēs:

- <http://validator.w3.org>
- <http://jigsaw.w3.org/css-validator>

Izstrādes procesā programmatūra tika regulāri testēta – pēc katrām veiktajām izmaiņām pirms koda pievienošanas repozitorijam, katras iterācijas beigās tika testēta jaunā izstrādātā funkcionalitāte, kā arī izstrādes beigās vēlreiz tika pārtestēta katra funkcija un visa izstrādātā programmatūra kopumā, lai pārliecinātos par izstrādātā produkta atbilstību izvirzītajām prasībām.

Regulāri ar pasūtītāju tika apspriesta arī lietotāju saskarne un ņemti vērā pasūtītāja ieteikumi uzlabojumiem, lai izstrādātais produkts būtu ērti lietojams un viegli saprotams.

Programmatūras prasību specifikācija un projektējuma apraksts tika izstrādāts un noformēts atbilstoši standartos izvirzītajām prasībām un norādījumiem. Katras dokumentācijas daļas ievada sadaļā “Saistība ar citiem dokumentiem” ir norādīti standarti, uz kuriem balstoties ir izstrādāta un noformēta attiecīgā dokumentācijas daļa.

Tika veikta arī programmatūras versiju uzturēšana, par ko sīkāk ir aprakstīts nākošajā nodaļā “6. Konfigurācijas pārvaldība”.

6. KONFIGURĀCIJAS PĀRVALDĪBA

Programmatūras izstrādes sākumā par vēlamo programmatūras versiju pārvaldības rīku tika noteikts SVN. Tas nodrošina programmatūras pirmkoda drošu versiju glabāšanu un aplūkošanu. Pirmkods repozitorijam tika pievienots gan pēc nelielu izmaiņu veikšanas un testēšanas, gan pēc katra lielāka vienuma izstrādes un testēšanas. Lai versiju kontrole būtu pārskatāmāka, katru reizi pievienojot pirmkodu repozitorijam tika pievienota arī ziņa ar Atlassian JIRA uzdevuma numuru (sk. “7. *Atlassian JIRA uzdevumu un problēmu pieteikumi*”).

Programmatūras būvējumu veidošanā tika izmantots uz Jenkins servera izveidots projekts, kuram kā datu avots tika uzstādīts programmatūras koda SVN repozitorijs.

Dokumentācijas versiju kontrole tika veikta izmantojot Microsoft OneDrive Office Online un WIKI, kas nodrošina gan ērtu versiju uzglabāšanu, gan apskati, gan dokumenta labošanu.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

7.1 Darbietilpības novērtējums - pirms un pēc

Tabulā attēlots salīdzinājums starp novērtējumu darbietilpībai, kas sastādīts pirms izstrādes uzsākšanas, un reālo izstrādei veltīto laiku.

tabula 7.1

Darbība	Paredzētais laiks	Reālais laiks	Starpība
Iepazīšanās ar jaunajām tehnoloģijām (SAPUI5)	10d (80h)	10d (80h)	0 h
XML modeļa attēlošana	10d (80h)	14d (112h)	- 32 h
Grunt uzdevumu konfigurācija	-	7d (56h)	-56 h
JSON modeļa attēlošana	7d (56h)	6d (48h)	+8 h
Visu modeļu saraksta attēlošana	7d (56h)	6d (48h)	+8h
Chrome DevTools integrācija	5d (40h)	4d (32h)	+8 h
Modeļu atjaunošana manuāli – ar pogas nospiešanu	3d (24h)	2d (16h)	+8h
Izpētīšana, kā paplašinājums sazinās ar tīmekļa vietnēm	5d (40h)	2d (24h)	+16 h
Izmaiņu uzraudzīšana modeļos un modeļu atjaunināšana automātiski	5d (40h)	4d (32 h)	+8 h
Testēšana	3d (24h)	3d (24h)	0 h
Dokumentācijas labošana un pabeigšana	5d (40h)	5d (40h)	0 h
Kopā:	60d (480h) = 3 personmēneši	63d (512h) = 3,2 personmēneši	

Reālais izstrādes laiks atšķīrās no plānotā šādu iemeslu dēļ:

- kaut arī pirmajās 2 nedēļās (80 h) cītīgi apguvu jaunu tehnoloģiju - SAPUI5 rīkkopu, iegūtās zināšanas nebija pilnībā pietiekamas, lai koda izstrāde veiktos tik raiti, kā biju plānojusi. Izstrādātais kods bija neprecīzs un to bija nepieciešams vairākkārt modificēt;
- netika ierēķināts pietiekami daudz laiks kļūdu labošanai – biju rēķinājusies, ka kļūdas būs, taču nebiju paredzējusi, ka to labošana būs tik laikietilpīga;

- skriptu rakstīšanu izmantojot Grunt automatizētai uzdevumu pildīšanai, nebiju ierēķinājusi vispār, taču galā tas aizņēma diezgan daudz laika, jo uzdevumus centos uzrakstīt pēc iespējas efektīvākus, lai veicot koda izstrādi man būtu jāveic pēc iespējas mazāk atkārtotu darbību manuāli.
- veicamie darbi nebija sadalīti pietiekami detalizēti, līdz ar to prognozēšana, cik laika katrs darbs varētu aizņemt, sanāca neprecīza. Kā piemēram, darbību Nr. 2 “XML modeļa attēlošana”, precīzākai laiktelpības aprēķināšanai varēja sadalīt apakšuzdevumos: XML modeļa datu iegūšana, iegūto datu parsēšana JSON formātā, jaunas kontroles izveide datu attēlošanai ar formatējumu, izvēršamu un sakļaujamu sadaļu implementēšana, testēšana, kļūdu labošana.
- šis bija pirmais šāda apjoma projekts un pieredzes trūkuma dēļ nebija iespējams novērtēt darbietilpību.

7.2 Eksperta metode

Darbietilpības novērtēšanā tika izmantota arī ekspertu metode – konsultējos ar kvalifikācijas darba vadītāju. Tā kā SAPUI5 ir jauna rīkkopa, kas ļauj būtiski saīsināt programmatūras kodu, tika nolemts, ka darbietilpības aprēķināšanai tādas tradicionālās metodes kā COCOMO, COCOMO II, nebūs piemērotas. Konsultējoties ar kvalifikācijas darba vadītāju, tika noskaidrots, ka cilvēkam ar pamatzināšanām par JavaScript, jQuery un bez pieredzes darbā ar SAPUI5 rīkkopu, projekta izstrāde aizņemtu apmēram 3 personmēnešus.

8. IZMANTOTĀS TEHNOLOĢIJAS UN RĪKI

Darba izstrādē tika izmantotas šādas tehnoloģijas un rīki:

- Programmēšanas valodas:
 - Javascript;
 - HTML5;
 - CSS3;
- jQuery bibliotēka;
- AJAX;
- SAPUI5 rīkkopa (skatīt “8.1 SAPUI5 rīkkopa”);
- Versiju kontroles rīks SVN (pieminēts “6. Konfigurācijas pārvaldība”);
- Nepārtrauktās integrācijas rīks – Jenkins (pieminēts “6. Konfigurācijas pārvaldība”);
- Projektu pārvaldības rīks Atlassian JIRA (pieminēts “6. Konfigurācijas pārvaldība” un “4. Projekta organizācija”);
- Koda kvalitātes pārvaldības rīks SonarQube (pieminēts “5. Kvalitātes nodrošināšana”);
- Uzdevumpildītājs Grunt (skatīt “8.2 Uzdevumpildītājs Grunt”);
- Izstrādes vide Visual Studio Code.

8.1 SAPUI5 rīkkopa

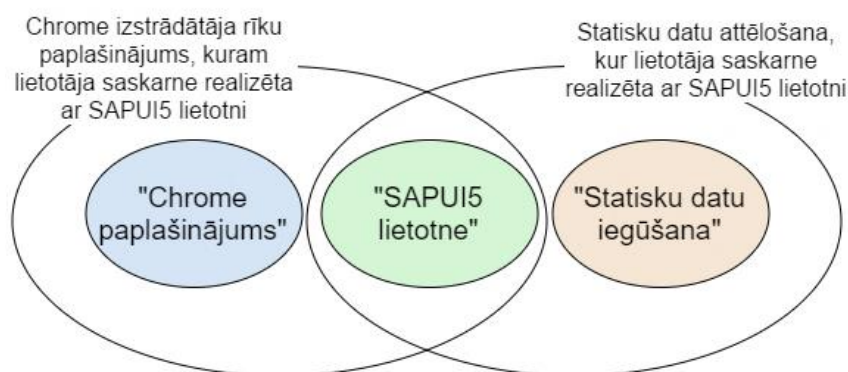
SAPUI5 ir rīkkopa, kas izstrādāta priekš HTML5. Lietotnes, kas ir izstrādātas izmantojot šo rīkkopu, ir reaģējošas uz dažādām pārlūkprogrammām un ierīcēm – viedtālruniem, planšetdatoriem un darbvirsām. Lietotāju saskarnes kontroles automātiski pielāgojas katras ierīces spējām.

SAPUI5 piedāvā šādus spēcīgus izstrādes konceptus:

- MVC pieeju;
- Datu saistīšanu;
- Iespējām bagātas lietotāja saskarnes kontroles;
- Datu iegūšanu no tādiem avotiem kā XML, JSON un oData, kā arī nodrošina modeli katram no iepriekš nosauktajiem formātiem;
- Modeļu piesaistīšana lietotāju saskarnes kontrolēm;
- Esošo kontroļu paplašināšanu.

8.2 Uzdevumpildītājs Grunt

Kā jau PPA nodaļā “2.2. Dekompozīcijas apraksts” tika skaidrots – izstrādātais paplašinājums ir sadalīts divās neatkarīgās daļās – “SAPUI5 lietotne” un “Chrome paplašinājums”. Projekta izstrādes procesā tika realizēta arī iespēja attēlot statistiskus datus, kurai ar paplašinājumu ir kopīga datu apstrādes un attēlošanas daļa “SAPUI5 lietotne”, taču atšķirīga ir datu iegūšanas daļa – dati tiek iegūti no statistiskiem modeļiem JSON un XML datnēs (sk. att. 8.1 Projekta *sadalījums*).



att. 8.1 Projekta *sadalījums*

Lai izstrādes laikā manuāli nebūtu jāveic atkārtotas darbības – kā datņu kopēšana, tika izmantots uzdevumpildītājs Grunt. Katru reizi kā izstrādes mapes datnēs tika veiktas izmaiņas, attiecīgās datnes tika automātiski pārkopētas uz gala mapēm (viena kā avots paplašinājumam un otra - statistiku datu attēlošanai).

Gunt tika izmantots ne tikai datņu kopēšanai, bet arī datņu sinhronizēšanai (“sync” uzdevums), datņu uzraudzīšanai (“watch” uzdevums), dzēšanai (“clean” uzdevums) LESS datņu kompilēšanai uz CSS (“less” uzdevums), SAPUI5 lietotnes datņu minimizēšanai (“openui5_preload” uzdevums).

9. PROGRAMMATŪRAS KODS

Pie programmatūras koda parauga izvēlējos likt fragmentu no kontrollera klases “Models” - metodi “onReloadAll”. Šī metode ir pirmais solis visu modeļu pārlādēšanā, metodē tiek izmantotas atzvanīšanas funkcijas, kā arī ir iespējams redzēt kā lietotāja saskarne tiek atjaunota mainot datus skatam piesaistītajā modelī.

No “Models” kontrollera klases iekļāvu arī metodi loadAllModels() – tajā var redzēt, kā tika realizēta iespēja metodes savienot “ķēdē” – gan izmantojot “Promise” objektus, gan atzvanīšanas funkcijas.

Iekļāvu arī koda fragmentu no “ChromeExtensionDataProvider” klases – metodi getModelNames(). Šīs metodes kods arī parāda “Promise” objekta izmantošanu, AJAX pieprasījumu veikšanu un chrome.devtools.inspectedWindow API izmantošanu.

9.1 Kontrollera klases “Models” metode “onReloadAll”

```
/**
 * Handles press on the "Reload all models" button
 * and is responsible for loading all the models for the first time if
 called from the onInit() method.
 * Calls method checkForSapui() and as parameters passes two callback
 functions, which are executed whether SAPUI5 is available or not.
 * If the SAPUI5 is available then method loadAllModels() is called.
 * @param {bool/object} init - Indicates whether this method is called from
 the onInit() method or from press on the "Reload all" button.
 * If this method will be called from onInit(), then this parameter will be
 boolean (true).
 * If this method will be called from press on the button "Reload all",
 then this parameter will be event object.
 */
onReloadAll: function(init) {

    var oModelsToolModel = sap.ui.getCore().getModel("Models");

    //checks if method for checking SAPUI5 availability is available
    if (sap.personas.devtools.dataProvider.checkForSapui) {

        //actions that will be completed if method for checking SAPUI5
        availability is available
        //checks if SAPUI5 is available

        sap.personas.devtools.dataProvider.checkForSapui(jQuery.proxy(function() {

            //actions that will be completed if SAPUI5 is available in
            context of controller object
            this.loadAllModels(true);
            oModelsToolModel.setProperty("/isVisibleNoData", false);
            oModelsToolModel.setProperty("/isVisibleTabBar", true);
        }, this), jQuery.proxy(function(init) {

            //actions that will be completed if SAPUI5 is not available in
            context of controller object
```



```

        if (init) {
            var oModelsToolModel = sap.ui.getCore().getModel("Models");
            oModelsToolModel.setProperty("/isVisibleNoData", true);
            oModelsToolModel.setProperty("/isVisibleTabBar", false);
        }

        sap.personas.devtools.errorHandler.displayErrorMessage("Couldn't
        load models!", "Couldn't load models, because SAPUI5 isn't
        available");
    }, this), true, init);
} else {

    //actions that will be completed if method for checking SAPUI5
    availability is not available
    this.loadAllModels(true);
    oModelsToolModel.setProperty("/isVisibleNoData", false);
    oModelsToolModel.setProperty("/isVisibleTabBar", true);
}
}
}

```

9.2 Kontroleru klases “Models” metode “loadAllModels”

```

/**
 * Gets all model names and types from inspected window by calling
 * getModelNames() method;
 * Then calls getModelData() for each model to get model data
 * @param {bool} bWatch - parameter that is passed to getModelData() method
 * (for detailed explanation look in ChromeExtensionDataProvider.js method
 * getModelData())
 */
loadAllModels: function(bWatch) {
    var that = this;

    //getModelNames() method returns promise object, when resolved, it
    returns object with model names and data
    sap.personas.devtools.dataProvider.getModelNames().then(function(data) {
        var aModelNames = Object.keys(data);
        for (var i = 0; i < aModelNames.length; i++) {

            sap.personas.devtools.dataProvider.getModelData(aModelNames[i],
            data[aModelNames[i]], jQuery.proxy(that.updateModel, that),
            bWatch);
        }
    });
}

```

9.3 Klases “ChromeExtensionDataProvider” metode “getModelNames”

```

/**
 * Gets all model names and types from inspected window using
 * chrome.devtools.inspectedWindow API
 * @returns {promise}
 */
getModelNames: function() {

    //acquires expression that will be evaluated in the inspected window
    from JavaScript file
    var jQueryPromise = $.ajax({ url: "eval/getModelNames.js", dataType:
    "text", data: "{}" });
    var promise = Promise.resolve(jQueryPromise)
    .then(function(sExpression) {

```

```

    return new Promise(function(resolve, reject) {

        //evaluates expression in the inspected window using
        chrome.devtools.inspectedWindow API
        //(expression returns an array with model names and types)
        chrome.devtools.inspectedWindow.eval(
            sExpression,
            function(result, isException) {
                if (isException) {

                    //handling errors, which may arise while
                    evaluating expression in the inspected window
                    sap.personas.devtools.errorHandler.chromeEvalErrorHandling(isException, "Models could not be loaded!");
                    reject(isException);
                } else {
                    resolve(result);
                }
            }
        );
    });
});
return promise;
}

```

9.4 Datne “getModelNames.js”

Iepriekšējās nodaļas (“9.3 Klases “ChromeExtensionDataProvider” metode “getModelNames””) koda paraugā tika veikts AJAX pieprasījums, lai iegūtu datus no JavaScript datnes “getModelNames.js”. Šajā nodaļā tiek aplūkots šīs datnes saturs - skripts, ko iegūst veicot AJAX pieprasījumu un tiek izpildīts inspicējamā logā izmantojot chrome.devtools.inspectedWindow API.

```

function getModelNames() {

    //Changes default getModel() method for Core object
    //If there is no parameter passed to getModel() method, then all models
    are returned
    //If there is parameter, then returns the specified model
    frames[0].sap.ui.core.Core.prototype.getModel = function(n) {
        if (n) {
            return this.oModels[n];
        } else {
            return this.oModels;
        }
    };

    //saves all XML and JSON model names and types in an object and returns
    it
    var oModelNames = {};
    var aKeys = Object.keys(frames[0].sap.ui.getCore().getModel());
    for (var i = 0; i < aKeys.length; i++) {
        if (frames[0].sap.ui.getCore().getModel(aKeys[i]) instanceof
frames[0].sap.ui.model.xml.XMLModel) {
            oModelNames[aKeys[i]] = "XML";
        }
    }
}

```

```

        if (frames[0].sap.ui.getCore().getModel(aKeys[i]) instanceof
frames[0].sap.ui.model.json.JSONModel) {
            oModelNames[aKeys[i]] = "JSON";
        }
    }
    return oModelNames;
}
getModelNames();

```

SECINĀJUMI

Projekta mērķis tika sasniegts - ir izstrādāts SAP Screen Personas 3.0 Chrome paplašinājums, kuru jau izmanto gala lietotāji - SAP Screen Personas 3.0 izstrādātāji.

Pēc lietotāju atsauksmēm tiek plānots arī paplašinājumu papildināt ar kādu no piedāvātajiem ieteikumiem – izveidot iespēju modeļus labot paplašinājumā un izmaiņas attēlot inspicējamā logā (SAP Screen Personas 3.0 Client), XML modeļa datus par veiktajām izmaiņām tēmā vai personalizētajā ekrānā attēlot atsevišķā logā un abus izmaiņu veidus nošķirt, šādi uzlabojot modeļa lasāmību un attēlot veiktās izmaiņas modeļos, tas ir, attēlot, kas izmainījies modelī kopš pēdējās ielādēšanas reizes.

Programmatūras izstrādes laikā nostiprināju zināšanas par JavaScript programmēšanas valodu, apguvu jaunu rīkkopu SAPUI5, guvu pieredzi projektu pārvaldīšanā izmantojot Atlassian JIRA, apguvu jaunu rīku SonarQube koda kvalitātes pārbaudei un iepazinos ar jaunu pirmkoda versiju kontroles rīku SVN.

Izstrādājot kodu, sapratu, cik liela nozīme ir koda komentēšanai, versiju kontrolei, kā arī darbietilpības novērtēšanai pirms projekta uzsākšanas, lai būtu iespējams noteikt, cik daudz laika projekta izstrāde aizņems un vai būs iespējams iekļauties noteiktajos termiņos.

Kopumā programmatūras izstrādes process bija interesants, jo tika apgūtas daudz jaunas tehnoloģijas, tika iegūtas zināšanas par programmatūras izstrādes procesiem un programmatūras izstrādes labajām praksēm.

IZMANTOTĀ LITERATŪRA

1. Akadēmiskā terminu datu bāze [tiešsaiste]. Pieejams: <http://termini.lza.lv/term.php> [Pārbaudīts 26.05.2016]
2. Chrome DevTools Extensions API [tiešsaiste]. Pieejams: <https://developer.chrome.com/extensions/devtools> [Pārbaudīts 26.05.2016]
3. Grunt documentation [tiešsaiste]. Pieejams: <http://gruntjs.com/getting-started> [Pārbaudīts 26.05.2016]
4. Ieteicamā prakse programmatūras projektējuma aprakstīšanai, LVS 72:1996, Latvijas Valsts standarts.
5. JavaScript reference documentation [tiešsaiste]. Pieejams: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference> [Pārbaudīts 26.05.2016]
6. jQuery API Documentation [tiešsaiste]. Pieejams: <http://api.jquery.com/> [Pārbaudīts 26.05.2016]
7. jQuery plugin for displaying JSON data [tiešsaiste]. Pieejams: <https://www.npmjs.com/package/jquery.json-viewer/> [Pārbaudīts 26.05.2016]
8. JSDoc [tiešsaiste] Pieejams: <http://usejsdoc.org/> [Pārbaudīts 26.05.2016]
9. Less documentation [tiešsaiste]. Pieejams: <http://lesscss.org/> [Pārbaudīts 26.05.2016]
10. M. Ķemme, "Programmētāju kvalifikācijas darbi 2012", 06.04.2012 [tiešsaiste]. Pieejams: <https://martinskemme.wordpress.com/2012/06/04/programmetaju-kvalifikacijas-darbi-2012/> [Pārbaudīts 26.05.2016]
11. Programmatūras prasību specifikācijas ceļvedis, LVS 68:1996, Latvijas Valsts standarts.
12. R. Simanovskis, "Rekomendācijas programmētāju kvalifikācijas darbiem", 06.14.2009 [tiešsaiste]. Pieejams: <https://elietas.wordpress.com/2009/06/14/rekomendacijas-programmetaju-kvalifikacijas-darbiem/> [Pārbaudīts 26.05.2016]
13. SAPUI5 [tiešsaiste]. Pieejams: <https://sapui5.netweaver.ondemand.com/sdk/> [Pārbaudīts 26.05.2016]

PIELIKUMI

1. Metodes “2.2.3.4.1 parse” ievaddatu un izvaddatu piemērs

XML formāta simbolu virkne, kas tiek padota metodei “2.2.3.4.1 parse”:

```
<ThemeEditor version="3.0"><GLOBAL_FLAGS></GLOBAL_FLAGS></ThemeEditor>
```

JSON formāta objekta uzbūve, ko atgriež metode “2.2.3.4.1 parse”:

```
lines [
  {
    num: "1",
    elements: [
      {
        type: "utility",
        text: "<"
      },
      {
        type: "element",
        text: "ThemeEditor"
      },
      {
        type: "utility",
        text: " "
      },
      {
        type: "attribute",
        text: "version"
      },
      {
        type: "utility",
        text: "="
      },
      {
        type: "attributeValue",
        text: "3.0"
      },
      {
        type: "utility",
        text: "\""
      },
      {
        type: "utility",
        text: ">"
      }
    ]
  },
  {
    num: "1.1",
    elements: [
      {
        type: "utility",
        text: "<"
      },
      {
        type: "element",
        text: "GLOBAL_FLAGS"
      },
    ]
  }
]
```

```

    {
      type: "utility",
      text: ">"
    },
    {
      type: "utility",
      text: "</"
    },
    {
      type: "element",
      text: "GLOBAL_FLAGS"
    },
    {
      type: "utility",
      text: ">"
    }
  ]
},
{
  num: "1end",
  elements: [
    {
      type: "utility",
      text: "</"
    },
    {
      type: "element",
      text: "ThemeEditor"
    },
    {
      type: "utility",
      text: ">"
    }
  ]
}
]

```

2.Skatam piesaistītā modeļa uzbūve

```

{
  models: [
    {
      name: "Employees",
      data: [2 items],
      type: "JSON",
      time: "Last loaded: 23/5/2016 @ 14:15:34"
    },
    {
      name: "Invoices",
      data: {1 item},
      type: "JSON",
      time: "Last loaded: 23/5/2016 @ 14:15:34"
    }
  ],
  modellist: [
    {
      name: "Employees",
      type: "JSON",
      position: 0
    },
  ],
}

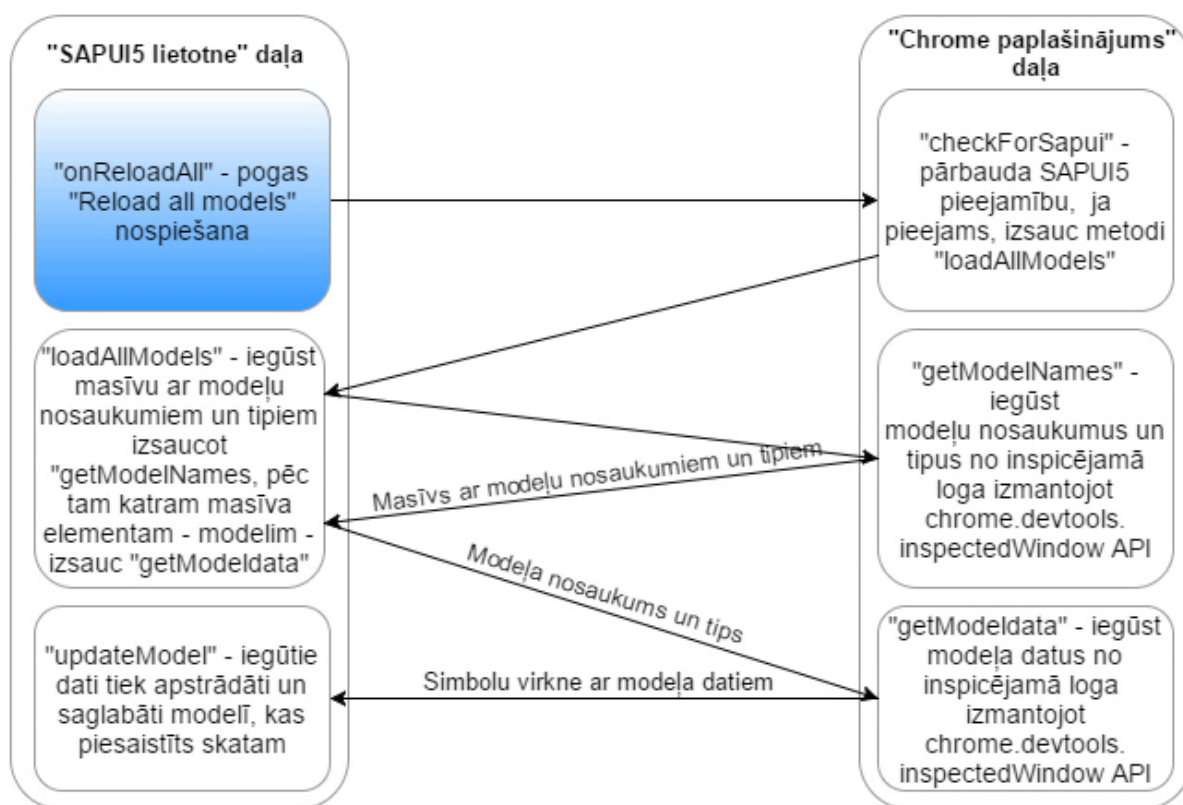
```

```

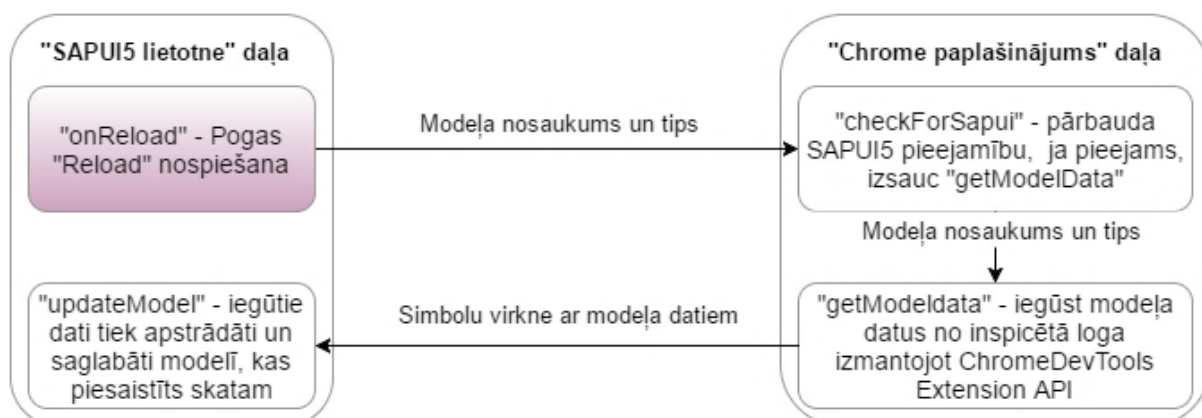
{
    name: "Invoices",
    type: "JSON",
    position: 1
},
isVisibleNoData: false,
isVisibleTabBar: true,
lastLoadedText: "Show last loaded",
lastLoadedIcon: "sap-icon://expand-group",
isVisibleLastLoaded: false,
messagePopoverButtonType: "Emphasized"
}

```

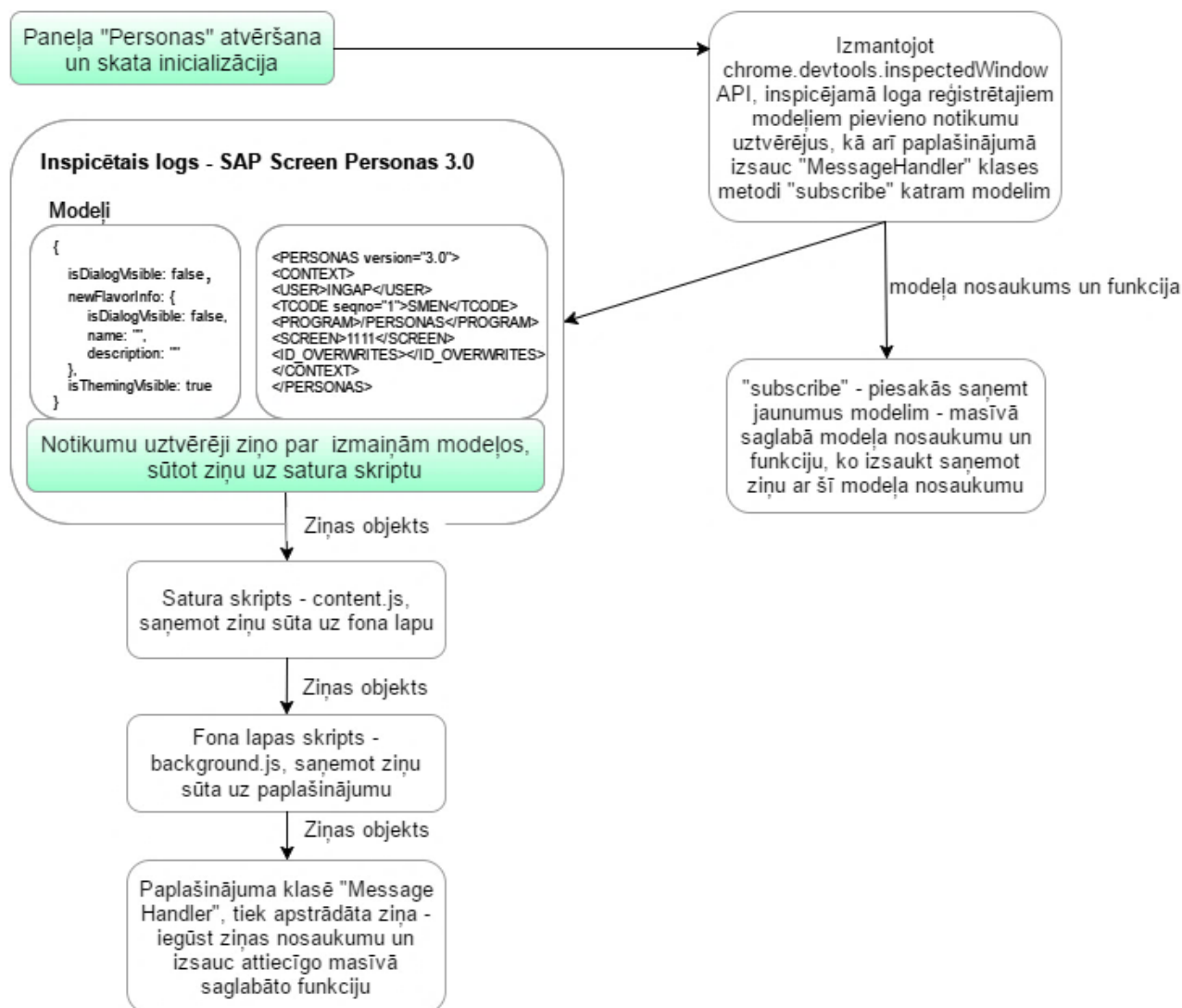
3. Prasības “1.3.3.2 Visu modeļu atjaunināšana” realizācijas grafisks attēlojums






4. Prasības “1.3.3.1 Modeļa atjaunināšana” realizācijas grafisks attēlojums



5. Prasības “1.3.3.3 Modeļa izmaiņu uzraudzīšana un datu atjaunošana automātiski realizācijas grafisks attēlojums



6. SonarQube ekrānuzņēmumi

	NAME	RULES COMPLIANCE	COVERAGE	UNIT TESTS	UNIT TESTS DURATION	COMPLEXITY	BUILD TIME	LINKS	COMPLEXITY /CLASS	DUPLICATED LINES (%)
★	 Personas3_DEV_Tools_JS	97.4% 				242 	16:36			0.0%

Dashboard

Hotspots

Reviews

Issues

Time Machine

TOOLS


Components

Issues Drilldown

Design

Libraries

Compare



Dashboards

Projects

Measures

Issues

Rules

Quality Profiles

Quality Gates

Inga Pire

Search

Personas3_DEV_Tools_JS

Version 1.0 - May 24 2016 16:36

Time changes...

Manage dashboards

Lines Of Code

1,300

JavaScript

Functions

101

Classes

0

Statements

601

Accessors

0

Files

23

Directories

8

Lines

1,768

Technical Debt

1d 5h

Issues

28

Blocker

0

Critical

0

Major

9

Minor

7

Info

12

Most Violated Components

More

Model.controller.js

100211

jsonViewer.js

100130

XMLParser.js

100110

Gruntfile.js

100110

ChromeExtensionDataProvider.js

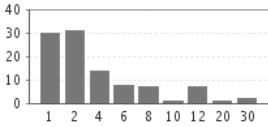
100111

Complexity

4.9/function

10.5/file

Total: 242



Functions

Files

Events

All

May 24 2016

Version

1.0

7. Atlassian JIRA uzdevumu un problēmu pieteikumi

The screenshot displays the Atlassian JIRA web application interface. At the top, a navigation bar includes the JIRA logo, a hamburger menu, and tabs for Dashboards, Projects, Issues, Gantt Chart, Agile, and a Create button. Below this, a sidebar on the left contains a 'FILTERS' section with a 'New filter' button and a list of filters: 'Find filters', 'My Open Issues', 'Reported by Me', 'Recently Viewed', and 'All Issues'. Below these are 'FAVOURITE FILTERS' with a note: 'You don't have any favourite filters.' The main content area is divided into a 'Search' section with a 'Save as' button and a 'Component: Personas DevTo...' dropdown, and a 'Details' section for the selected issue. The issue is titled 'XML view' and is part of the 'TRAINING-128' project. The 'Details' section shows the issue's type as 'Task sub-task', priority as 'Major', and status as 'CLOSED'. It also lists the version as '2016-P-UI5-plugin' and the component as '2016-Personas-interns, ... (1)'. The 'Description' section provides a detailed explanation of the task: 'Implement XML view rendering, which uses XML structure generated by XML parser. XML nodes without children are displayed in a single line. XML nodes with children are displayed in 2+x lines, where first line is opening tag with attributes, followed by children node rendered content and ending with closing tag. XML elements, attributes, attribute values and inner content is styled differently.' The 'Issue Links' section shows a dependency on 'TRAINING-127 XML parser', which is also 'CLOSED'. The 'Activity' section at the bottom includes tabs for 'All', 'Comments', 'Work Log', 'History', 'Activity', 'CI Builds', and 'Subversion', with 'Comments' currently selected.

Search Save as

Training Type: All Status: All Assignee: All Contains text More Advanced

Component: Personas DevTo...

Filters

New filter

Find filters

My Open Issues

Reported by Me

Recently Viewed

All Issues

FAVOURITE FILTERS

You don't have any favourite filters.

TRAINING-210
Throttle Model Updates

TRAINING-179
Model usability improvements

TRAINING-170
Multiple Model Support

TRAINING-171
UI for multiple models

TRAINING-172
Add JSON model support

TRAINING-173
Model content loads very slow

TRAINING-163
XML model reload

TRAINING-158
Configure Chrome extension build u...

TRAINING-126
XML Viewer

TRAINING-128
XML view

TRAINING-127
XML parser

TRAINING-129
Mock Data Provider

TRAINING-125

XML view

Edit Comment Assign More

Details

Type: Task sub-task Status: **CLOSED** (View Workflow)

Priority: Major

Affects Version/s: 2016-P-UI5-plugin Resolution: Fixed

Component/s: 2016-Personas-interns, ... (1) Fix Version/s: 2016-P-UI5-plugin

Labels: None

Description

Implement XML view rendering, which uses XML structure generated by XML parser. XML nodes without children are displayed in a single line. XML nodes with children are displayed in 2+x lines, where first line is opening tag with attributes, followed by children node rendered content and ending with closing tag. XML elements, attributes, attribute values and inner content is styled differently.

Issue Links

depends on TRAINING-127 XML parser **CLOSED**

Activity

All **Comments** Work Log History Activity CI Builds Subversion

Mercurial Commits

Edit

Comment

Assign

More

Clarified

Activity

All	Comments	Work Log	History	Activity	CI Builds	Subversion	Mercurial Commits
-----	----------	----------	---------	----------	-----------	-------------------	-------------------

Repository	Revision	Date	User	Message
Training	#1328	Tue May 03 17:57:49 EEST 2016	ipire	<div>TRAINING-183</div> <div>Files Changed</div> <div> <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/controller/Model.controller.js <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/view/Model.view.xml <div>MODIFY</div> /personas/personas3-dev-tools/src/mock/init.js <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/controller/BaseController.js <div>MODIFY</div> /personas/personas3-dev-tools/src/chrome-extension/init.js <div>ADD</div> /personas/personas3-dev-tools/src/dev-tools/ErrorHandler.js </div>
Repository	Revision	Date	User	Message
Training	#1335	Thu May 05 14:58:18 EEST 2016	ipire	<div>TRAINING-183</div> <div>Files Changed</div> <div> <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/controller/Model.controller.js <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/view/Model.view.xml <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/view/JSONModel.fragment.xml <div>MODIFY</div> /personas/personas3-dev-tools/src/mock/init.js <div>MODIFY</div> /personas/personas3-dev-tools/src/dev-tools/controller/BaseController.js </div>

8. JSDoc

8.1 Kontrolleru klase “Models”

getDate(bOnlyDate_{opt}) → {string}

Returns current date and time

Parameters:

Name	Type	Attributes	Description
bOnlyDate	bool	<optional>	Indicates whether to return only date or date with text "Last loaded:"

Source: Model.controller.js, line 312

Returns:

only date or date with text "Last loaded"

Type string

handleMessagePopoverPress(oEvent)

Handles press on the message popover button - opens message popover

Parameters:

Name	Type	Description
oEvent	object	Event object

Source: Model.controller.js, line 125

loadAllModels(bWatch_{opt})

Gets all model names and types from the inspected window by calling getModelNames() method, which returns promise object. When promise is resolved, array with model names and types is returned. Then calls getModelData() for each model to get model data and as parameters passes model name, type and callback function - updateModel().

Parameters:

Name	Type	Attributes	Description
bWatch	bool	<optional>	Parameter that is passed to getModelData() method (for detailed explanation look in ChromeExtensionDataProvider.js method getModelData())

Source: Model.controller.js, line 111

modellistFactory(sId, oContext) → {object}

Creates controls from model data. Called for each list entry to create the controls necessary to represent the current entry.

Parameters:

Name	Type	Description
sId	string	Unique identifier for created control
oContext	object	Context object

Source: Model.controller.js, line 185

Returns:

Depending on the model type, XML or JSON fragment control

Type object

onCollapseAllJson()

Handles press on button "Collapse all" in JSON model view - collapses all sections in JSON model.

Source: Model.controller.js, line 344

onCollapseAllXml()

Handles press on button "Collapse all" in XML model view - collapses all sections in XML model.

Source: Model.controller.js, line 330

onExpandAllJson()

Handles press on button "Expand all" in JSON model view - expands all sections in JSON model.

Source: Model.controller.js, line 358

onExpandAllXml()

Handles press on button "Expand all" in XML model view - expands all sections in XML model.

Source: Model.controller.js, line 337

onInit()

Called when a view is instantiated and its controls (if available) have already been created. Used to modify the view before it is displayed to bind event handlers and do other one-time initialization. This method: creates an item template for message popover, creates a message popover control, binds model to view, binds model to message popover control, sets up message popover - adds some events and buttons to default control, loads model data for the first time, and sets up properties that are bound to view.

Source: Model.controller.js, line 30

onReload(oEvent)

Handles press on "Reload" button. Gets active model name and type from context, then calls method checkforSapui() and as parameters passes two callback functions, which are executed whether SAPUI5 is available or not. If SAPUI5 is available then calls callback function - method getModelData() to get model data and as parameters passes model name, type and callback function - updateModel().

Parameters:

Name	Type	Description
oEvent	object	Event object

Source: Model.controller.js, line 213

onReloadAll(init)

Handles press on the "Reload all models" button and is responsible for loading all the models for the first time if called from the onInit() method. Calls method checkForSapui() and as parameters passes two callback functions, which are executed whether SAPUI5 is available or not. If the SAPUI5 is available then callback function - method loadAllModels() is called.

Parameters:

Name	Type	Description
init	bool/object	Indicates whether this method is called from the onInit() method or from press on the "Reload all" button. If this method will be called from onInit(), then this parameter will be boolean (true). If this method will be called from press on the button "Reload all", then this parameter will be event object.

Source: Model.controller.js, line 138

onShowLastLoaded()

Handles press on the "Show last loaded"/"Hide last loaded" button. Shows or hides the sidebar and changes text/icon on the button.

Source: Model.controller.js, line 165

updateModel(sData, sModel, sType)

Parses and updates model data. If there is an error while parsing, it is processed and corresponding error message is displayed. If parsing is done without errors, then updates model data in model that is bound to view and adds model name and reload time to the list in sidebar.

Parameters:

Name	Type	Description
sData	string	Model data
sModel	string	Model name
sType	string	Model type

Source: Model.controller.js, line 239

8.2 Kontroles klase “XMLViewer”

onAfterRendering()

After rendering XMLViewer control, adds "click" event listeners to line numbers. When clicking on line numbers sections are expanded or collapsed.

Source: XMLViewer.js, line 80

renderer(oRM, oControl)

Renders XMLViewer control with the needed formatting.

Parameters:

Name	Type	Description
oRM	object	Renderer manager
oControl	object	Control object

Source: XMLViewer.js, line 31

8.3 Kontroles klase “JSONViewer”

onAfterRendering()

After rendering JSONViewer control, its data is formatted using "jsonViewer" plugin (found in ./jsonViewer/jsonViewer.js)

Source: JSONViewer.js, line 33

renderer(oRM, oControl)

Renders JSONViewer control

Parameters:

Name	Type	Description
oRM	object	Renderer manager

Name	Type	Description
oControl	object	Control object

Source: JSONViewer.js, line 22

8.4 Klase “XMLParser”

parse(root, key, personas) → {object}

Parses XML string to JSON object (recursively)

Parameters:

Name	Type	Description
root	string/object	At the first call of this method parameter "root" is XML string, if the method is called recursively then it is object - XML DOM node.
key	string	Line number. If the method is called for the first time, the key will be "1".
personas	object	Object in which data about each XML DOM node is stored.

Source: XMLParser.js, line 16

Returns:

JSON object

Type object

8.5 Klase “ChromeExtensionDataProvider”

checkForSapui(fnCallbackTrue, fnCallbackFalse, bSetInterval_{opt})

Checks if SAPUI5 is available in the inspected window using chrome.devtools.inspectedWindow API.

Calls corresponding callback function. If SAPUI5 isn't available, sets interval on this method to check SAPUI5 availability each second. When SAPUI5 is available, removes interval.

Parameters:

Name	Type	Attributes	Description
fnCallbackTrue	function		Callback function that is called if SAPUI5 has loaded.
fnCallbackFalse	function		Callback function that is called if SAPUI5 hasn't loaded.
bSetInterval	bool	<optional>	Indicates whether to set interval on this function if SAPUI5 isn't available.

Source: ChromeExtensionDataProvider.js, line 125

getModelData(sModel, sType, fnCallback, bWatch_{opt})

Gets model data from inspected window using chrome.devtools.inspectedWindow API. Also performs AJAX request to acquire expression from JavaScript file that will be evaluated in the inspected window. If bWatch parameter is passed then also subscribes to receive news for this model (by calling subscribe() method) and attaches function to this models change event in the inspected window.

Parameters:

Name	Type	Attributes	Description
sModel	string		Model names
sType	string		Model type
fnCallback	function		Callback function that handles acquired model data
bWatch	bool	<optional>	Indicates whether to subscribe for this models news (by calling subscribe() method) and attach function to change event in the inspected window

Source: ChromeExtensionDataProvider.js, line 26

getModelNames() → {promise}

Gets all model names and types from inspected window using chrome.devtools.inspectedWindow API. Also performs AJAX request to acquire expression from JavaScript file that will be evaluated in the inspected window.

Source: ChromeExtensionDataProvider.js, line 88

Returns:

Type promise

8.6 Klase “MessageHandler”

subscribe(sKey, sFrom, fnCallback)

Subscribes to execute a specific function, when message with the specified key is received

Parameters:

Name	Type	Description
sKey	string	a key, that will be saved in "subscriptions" object
sFrom	string	from where the function was called
fnCallback	function	function, that will be executed when message with key "sKey" will be received

Source: MessageHandler.js, line 41

unsubscribe(sKey, sFrom)

Unsubscribes to execute a specific function, when message with the specified key is received

Parameters:

Name	Type	Description
sKey	string	a key, that is saved in "subscriptions" object
sFrom	string	from where the function was called

Source: MessageHandler.js, line 53

9. User guide

1. Introduction

1.1 Intended readership

This document covers the use of SAP Screen Personas 3.0 Chrome developer tools extension and it is meant for its users - SAP Screen Personas 3.0 developers.

1.2 Applicability

This Software User Documentation (SUD) applies to the SAP Screen Personas 3.0 Chrome developer tools extension, version 1.0.

1.3 Purpose

The purpose of this SUD is to assist the user in installing and using the SAP Screen Personas 3.0 Chrome developer tools extension.

1.4 How to use this document

Document is divided in chapters:

- Chapter 1 gives introduction to this document;
- Chapter 2 gives an overview of the SAP Screen Personas 3.0 Chrome developer tools extension;
- Chapter 3 contains tutorials for common tasks that enable users to get started quickly.

2. Overview

SAP Screen Personas 3.0 Chrome Developer Tools extension is a software component, that allows to add an additional functionality to Google Chrome browser. Extension adds a new panel “Personas” to Google Chrome Developer Tools window, evaluates code in the

inspected window to get the needed information and displays acquired information in the panel.

The goal of extension is to simplify SAP Screen Personas 3.0 debugging and troubleshooting process by presenting current client application state in a clear and readable manner. The extension is meant for a specific user group – SAP Screen Personas 3.0 developers.

3. Tutorial

3.1 Adding SAP Screen Personas 3.0 extension to Google Chrome

This tutorial will guide you through installation process of SAP Screen Personas 3.0 extension in five steps.

1. First of all, acquire the source code from SVN.
2. Open Google Chrome browser, click menu ≡ at the top-right of your browser window, navigate to More tools >Extensions or just type in the search field “chrome://extensions”.
3. Then look for the option “Developer mode” and enable it. Once you’ve enabled the “Developer mode” option, it will modify the extensions tab and reveal “Load unpacked extension” option.
4. Now simply click on the “Load unpacked extension” option, it will ask you for the directory from where the extension is to be loaded. Here specify the directory and click on OK.
5. That’s it, extension will now be added to Google Chrome, now scroll down in the “Extensions” tab, you’ll see SAP Screen Personas 3.0 Chrome DevTools extension in the list. Now make sure that option “Enabled” is checked and start using extension!

3.2 Removing SAP Screen Personas 3.0 extension from Google Chrome

If for some reason you have decided that you don’t want to use this extension anymore, you can remove it by simply completing these steps.

To remove an extension from Google Chrome:

1. Open Google Chrome browser, click menu ≡ at the top-right of your browser window, navigate to More tools >Extensions or just type in the search field “chrome://extensions”.
2. On the extension you want to remove, click Remove from Chrome.

3. A notice to remove the extension will appear. Click Remove.

Or:

1. If an extension has an icon in your Chrome toolbar, you can right-click on the icon and select Remove from Chrome to uninstall the extension.

3.3 Using SAP Screen Personas 3.0 extension

This tutorial will show you how to start using SAP Screen Personas 3.0 extension after completing all steps in section “2.1 Adding SAP Screen Personas 3.0 extension to Google Chrome”.

Open in Google Chrome browser SAP Screen Personas 3.0 Client and open DevTools Window. To access the DevTools, open a web page or web app in Google Chrome and then either:

- Select the menu \equiv at the top-right of your browser window, then select Tools > Developer Tools.
- Use Ctrl+Shift+I (or Cmd+Opt+I on Mac) to open the DevTools.

After that, look for panel “Personas” and click on it. You should see a similar screen as in the picture below. Currently SAPUI5 hasn’t loaded yet and there is no model to display.

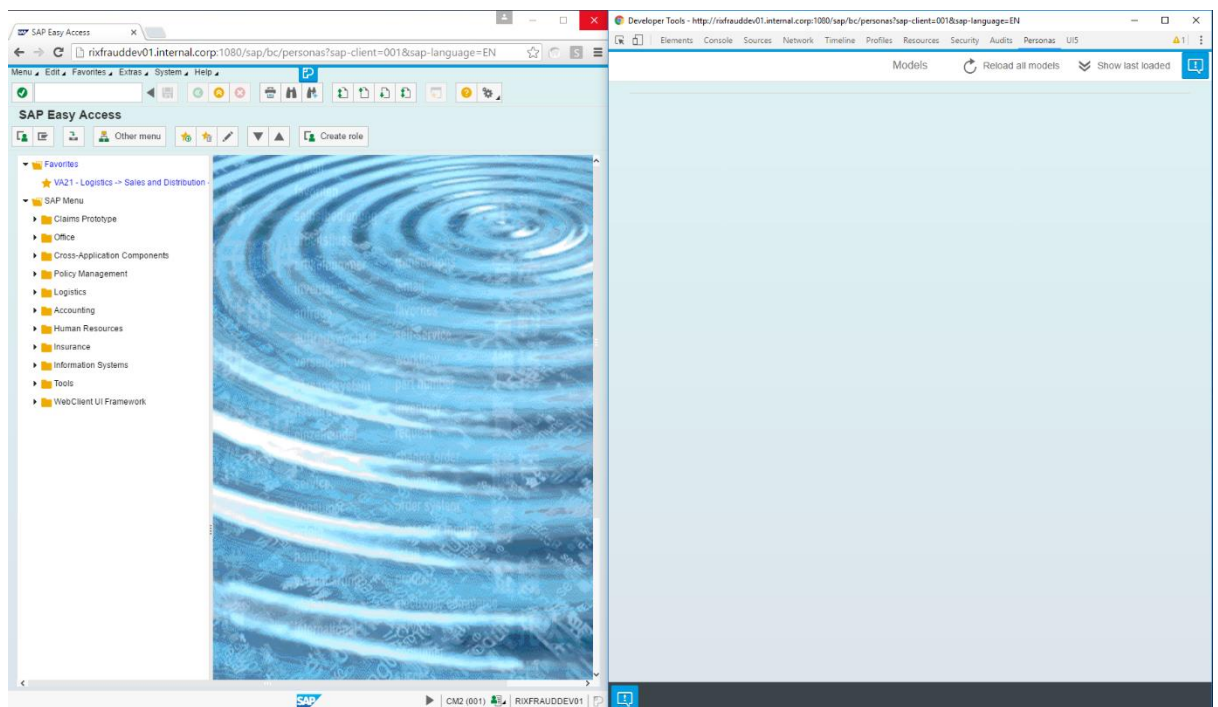


Figure 3.1 **Personas 3.0 Client and DevTools window**

SAPUI5 is loaded for the first time when you click on the button with letter “P” and the toolbar is shown as in the image below.

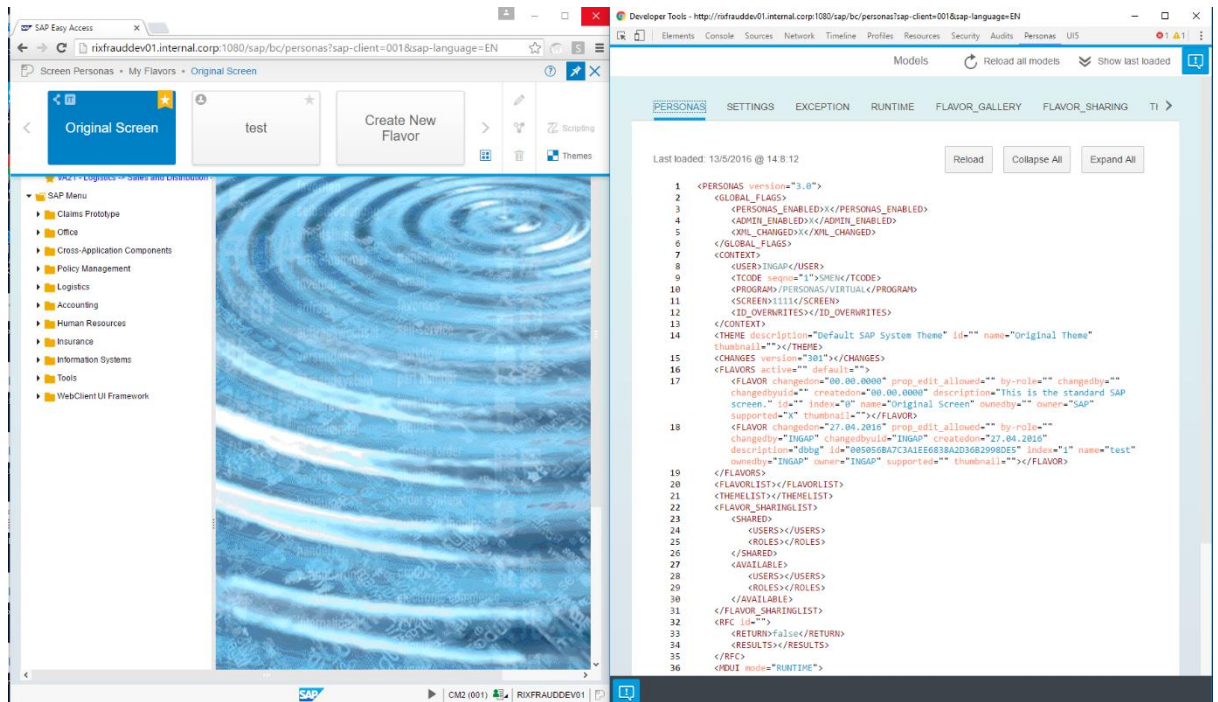


Figure 3.2 Personas 3.0 Client when SAPUI5 has loaded and DevTools window

After completing these steps, you should see a tab bar with all models.

In this extension you can:

- navigate through models by clicking on tabs;
- reload all models by pressing “Reload all models” button in the page header;
- show or hide list with last fifteen loaded models by pressing “Show last loaded” button in the page header and hide it by clicking “Hide last loaded”;
- view error, warning, success and information messages by clicking on the info button in the page header;
- models should reload automatically when change happens, but in case of error you can reload model manually by clicking “Reload” button;
- expand all sections by clicking “Expand all” and collapse all sections by clicking “Collapse all”;
- expand and collapse sections that you want by clicking on the bolded line numbers in XML view and clicking on grey triangles in JSON view.

DOKUMENTĀRĀ LAPA

Kvalifikācijas darbs „*SAP Screen Personas 3.0 Chrome izstrādātāja rīku paplašinājuma 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: *Inga Pīre* _____ .05.2016.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs/a: *Dipl. mat. Imants Pops* _____ .05.2016.

Recenzents: *M. dat. Jānis Baiža*

Darbs iesniegts 30.05.2016.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova* _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

____.06.2016. prot. Nr. _____

Komisijas sekretārs(-e): _____