LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

Informatīvā paneļa "InfoDesk" konfigurējamu logrīku izstrāde

KVALIFIKĀCIJAS DARBS

Autors: **Kaspars Upenieks**

Studenta apliecības Nr.: ku13004

Darba vadītājs: Dr.inž. Artis Teilāns

RĪGA 2015

ANOTĀCIJA

Kaspara Upenieka kvalifikācijas darbā " Informatīvā paneļa "InfoDesk" konfigurējamu logrīku izstrāde" ir pilnveidoti 4 logrīki un izstrādāts viens papildus logrīks. Logrīki kuriem veikti būtiski uzlabojumi ir 'WeatherWidget', 'VardaDienuWidget', 'DateWidget' un 'JiraReaderWidget', bet papildus izstrādātais logrīks ir 'TextEditorWidget'. Logrīku mērķis - atvieglot informatīvā paneļa „Infodesk" lietotājiem darbu ar šo lietotni un nodrošināt papildus funkcionalitāti. Galvenie šo logrīku uzdevumi ir sekojoši:

- 'WeatherWidget':sniedz lietotājam datus par pašreizējo gaisa temperatūru;
- 'VardadienuWidget': sniedz lietotājiem datus par konkrētās dienas vārda dienu gaviļniekiem;
- 'DateWidget': uzrāda lietotājiem datumu un mēnesi;
- 'JiraReaderWidget': uzrāda lietotājam konkrētās dienas citātu jeb 'domu graudu', nolasot to no citātu saraksta;
- 'TextEditorWidget': ļauj lietotājam ievadīt tekstu ar paša izvēlētu teksta formatējumu.

Visi logrīki ir izstrādātas Java programmēšanas valodā, izmantojot Android operētājsistēmas iebūvētās bibliotēkas.

Atslēgas vārdi: Android, Agile, Logrīki, InfoDesk, WeatherWidget, VardadienuWidget, DateWidget, JiraReaderWidget, TextEditorWidget

ABSTRACT

Qualification work " Configurable widget development for informative panel "InfoDesk" ",which is accomplished by Kaspars Upenieks contains 4 widgets with improved functionality and one extra widget.Widgets with significant improvements are 'WeatherWidget', 'VardaDienuWidget', 'DateWidget' and 'JiraReaderWidget', but additionally developed widget is 'TextEditorWidget'. Widget aim - to facilitate information panel "Infodesk" users work with this software application and provide additional functionality. The main tasks of all widgets are as follows:

• 'WeatherWidget': gives data on the current air temperature;
• 'VardadienuWidget': provides users with data of calendar names on a particular day;
• 'DateWidget': shows users date and month of year;
• 'JiraReaderWidget': displays the user the daily quote or 'grain of thought' by reading it from a quote list;
• 'TextEditorWidget': allows users to enter text with custom formatting.

All widgets have been developed in the Java programming language using the Android operating system built-in libraries.

Keywords: Android, Agile, Widgets, InfoDesk, WeatherWidget, VardadienuWidget, DateWidget, JiraReaderWidget, TextEditorWidget

Saturs

1. IEVADS

1.1. Nolūks

Programmatūras prasību specifikācijas nolūks ir aprakstīt visu 5 Android logrīku prasības. Dokuments ir paredzēts logrīku projektētājam un pasūtītājam produkta funkcionalitātes izveidei un prasību saskaņošanai.

1.2. Darbības sfēra

Šajā dokumentā ir definētas prasības 4 Android logrīku uzlabošanai un 1 logrīka izveidei. Programmatūras produkta mērķis ir pievienot papildus funkcionalitāti jau esošajiem logrīkiem un izveidot jaunus logrīkus to nepieciešamības gadījumā, tādējādi atvieglojot lietotāja darbu ar informatīvo paneli „InfoDesk".

Nedaudz par „InfoDesk" lietotni:

Tā kā Windows vidē ir pieejams prezentācijas un attēlošanas rīks "PowerPoint", ir nepieciešamība izveidot analogu rīku "InfoDesk" Android vidē, kas spēj izveidot un atskaņot prezentācijas. Ņemot vērā mūsdienu ierīču tehnoloģiskās iespējas tika pieņemts lēmums šo rīku izveidot Android operētājsistēmas versijām sākot no 4.0 jeb API 14. Lai "InfoDesk" padarītu konkurētspējīgāku, tam tika pievienota iespēja imitēt Android operētājsistēmas sākumekrānu t.i. slaidiem pievienot logrīkus, kā arī atskaņot video, interaktīvi atainot tiešsaistes vietnes un pievienot dažādus paplašinājumus, papildus tam visam prezentācijas var augšupielādēt un lejupielādēt, izmantojot Google Drive sistēmu, kā arī uzstādīt tās automātisku atjaunināšanu.(7.avots) InfoDesk lietotnes slaidu pievienošanas un rediģēšanas skati redzami attēlā 1.2.1.

1.2.1. att. **”InfoDesk” lietotnes slaidu pievienošanas un rediģēšanas skati**

1.3. Definīcijas, akronīmi un saīsinājumi

Android	Mobilo ierīču operētājsistēma.
Android Studio	Android lietotņu izstrādes vide.
Intent	Viens no Android operētājsistēmas interfeisiem, kuru izmanto, lai veidot starp-programmatūru saziņu.
Java	Programmēšanas valoda.
Mercurial	Versiju kontroles sistēma.
SourceTree	Versiju kontroles sistēmas pārvaldības grafiskās saskarnes rīks.
XML	Paplašināmā iezīmēšanas valoda.

2. VISPĀRĒJAIS APRAKSTS

2.1. Produkta perspektīva

Visi izstrādātie Android logrīki ir funkcionēt spējīgi kā asevišķas lietotnes, tāpēc jebkura Android ierīce tos var brīvi lietot arī ārpus „InfoDesk",piemēram, novietojot tos uz mobilās ierīces sākuma ekrāna.

2.2. Produkta funkcijas

Tālāk aprakstīts kādas funkcijas jāpilda katram logrīkam.

2.2.1. WeatherWidget logrīks

Logrīkam jānolasa no servera dati, tie jāapstrādā un jāattēlo konkrētā brīža gaisa temperatūra un balstoties uz laikapstākļu datiem(apmācies,saulains,lietains laiks) jāattēlo atbilstoša ikona.

2.2.2. VardadienuWidget logrīks

Logrīkam jānolasa no servera dati,tie jāapstrādā un jāattēlo konkrētās dienas vārdadienu gaviļnieki.

2.2.3. DateWidget logrīks

Logrīkam jāattēlo konkrētās dienas datums un mēnesis, nolasot to no iekārtas uzstādījumiem.

2.2.4. JiraReaderWidget logrīks

Logrīkam jānolasa no servera dati, tie jāapstrādā un jāattēlo citāts jeb 'domu grauds'.

2.2.5. TextEditorWidget logrīks

Logrīkam jādod iespēja lietotājam ievadīt paša izvēlētu tekstu ar atbilstošu formatējumu.

2.3. Lietotāja raksturiezīmes

Skolu audzēkņi, uzņēmumu un iestāžu darbinieki vai citi lietotāji, kuri ikdienā izvēlas lietot informatīvo paneli „InfoDesk" vai kādu no logrīkiem. Lietotājam nav jābūt ar īpašām priekšzināšanām.

2.4. Vispārējie ierobežojumi

Lai lietotu visus logrīkus lietotājam ir nepieciešama Android ierīce, kuras versija ir sākot no 4.0.3. vai augstāka.

2.5. Pieņēmumi un atkarības

Tā kā WeatherWidget logrīka datu serveris ļauj nolasīt datus no servera tikai 1000 reizes dienā tika pieņemts veikt datu atjaunināšanu šim logrīkam tikai reizi stundā.

3. LIETOTĀJA STĀSTI

Programmatūra tika izstrādāta pielietojot spējās programmēšanas metodoloģijas pieeju, tāpēc funkcionālās prasības tiek noteiktas balstoties uz „lietotāju stāstiem", kuri radās programmatūras izstrādes laikā.

Balstoties uz „lietotāju stāstiem" izstrādātājs varēja noteikt ko vēlas lietotājs un kāda ir katras prasības prioritāte attiecībā pret citām prasībām.

„Lietotāja stāsts" ir izveidots tabulas veidā, kurā ir attēloti visi lietotāja stāsti. Šie stāsti ir sakārtoti noteiktā secībā, balstoties uz to, kāda ir katra konkrētā lietotāja stāsta prioritāte un sarežģītība, kas izteikta punktos, kurus izstrādātājs noteica balstoties uz savām spējām zināšanām, par katras konkrētās prasības realizāciju.

3.1. Lietotāju stāsti sadalīti divās daļās

Lai lietotāju stāstus būtu vieglāk uztvert un dokumentēt konkrētās prasībās, tie tika sadalīti 2 daļās: logrīku uzlabojumu lietotājstāsti un 'TextEditorWidget' logrīka izveides lietotājstāsti.

Vispārējie logrīku uzlabojumu lietotājstāsti ir attēloti tabulā 3.1.1.

3.1.1. tabula

Vispārējie logrīku uzlabojumu stāsti

Lietotāja stāsts	Prioritāte	Sarežģītības punkti
1.Kā lietotājs, es vēlos, lai visiem logrīkiem ir iespējams iestatīt teksta krāsu un burtu izmēru.	1	13
2.Kā lietotājs, es vēlos, lai visiem logrīkiem jebkurā brīdī var nomainīt teksta krāsu un burtu izmēru.	2	5
3.Kā lietotājs, es vēlos, lai logrīkos vienmēr tiktu uzrādīta jaunākā informācija.	2	5
4.Kā lietotājs, es vēlos, lai	3	2

visi logrīki darbotos, ne tikai ‘InfoDesk’ lietotnē, bet arī uz sākuma ekrāna.		
5.Kā lietotājs, es vēlos, lai katram vienāda tipa logrīkam var uzstādīt atšķirīgus uzstādījumus.	4	5
6.Kā lietotājs, es vēlos, lai ‘JiraReaderWidget’ logrīkam, katru dienu automātiski mainītos citāti.	4	8
7.Kā lietotājs, es vēlos, lai ‘DateWidget’ logrīku būtu iespējams attēlot ar mazajiem vai lielajiem burtiem.	5	3
8.Kā lietotājs, es vēlos, lai uzstādītās logrīku knfigurācijas varētu pārnest uz citu ierīci.	5	8

‘TextEditor’ logrīka izveides lietotājstāsti ir attēloti tabulā 3.1.2.

3.1.2.tabula

‘TextEditor’ logrīka izveides stāsti

Lietotāja stāsts	Prioritāte	Sarežģītības punkti
9.Kā lietotājs, es vēlos, lai tiktu izveidots logrīks, kurš ļauj man uzlikt uz ekrāna ievadīto tekstu.	1	21
10.Kā lietotājs, es vēlos, lai ievadīto tekstu es jebkurā laikā varu nomainīt.	2	5

11.Kā lietotājs, es vēlos, lai tekstam var nomainīt krāsu un izmēru.	2	8
12.Kā lietotājs, es vēlos, lai tekstu var rakstīt slīprakstā, treknrakstā, ar pasvītrojumu, vai ar pārsvītrojumu.	3	8
13.Kā lietotājs, es vēlos, lai šis logrīks ļauj atsevišķi apstrādāt katru teksta simbolu.	3	13

3.2. Lietotāju stāstu sadalījums pa iterācijām

Lietotājstāstu sadalījumi pa iterācijām attēloti tabulās. Katra tabula sastāv no divām daļām: mērķa un uzdevuma. Mērķis raksturo klienta prasības, bet uzdevuma sadaļa rakstuto to, kā izpildītājs plāno šo prasību realizēt.

Pirmās iterācijas lietotājstāsts ir attēlots tabulā 3.2.1.

3.2.1.tabula

Pirmā iterācija

Lietotāja stāsts	**Uzdevums**
1.Kā lietotājs, es vēlos, lai visiem logrīkiem ir iespējams iestatīt teksta krāsu un burtu izmēru.	1.1.Pievienot visiem logrīkiem konfigurācijas aktivitāti, kura ļaus lietotājam norādīt vēlamos iestatījumus.
	1.2.Izveidot XML dizaina failu konfigurācijas aktivitātes grafiskajai saskarnei.
	1.3.Izveidos aktivitāti, kas saglabās lietotāja ievadītos iestatījumus XML formāta failos.
	1.4.Izveidot metodi, kas iestatīs lietotāja ievadītos datus konkrētajam logrīkam.
2.Kā lietotājs, es vēlos, lai visiem	2.1.Izveidot izsaukumu, kurš lietotājam

logrīkiem jebkurā brīdī var nomainīt teksta krāsu un burtu izmēru.	pieskaroties logrīka grafiskajai saskarnei atkārtoti izsauks logrīka konfigurācijas aktivitāti.
	2.2.Pievienot konfigurācijas aktivitātei trigeri, kurš saņemot izsaukumu atvērs konfigurācijas aktivitāti.
4.Kā lietotājs, es vēlos, lai visi logrīki darbotos, ne tikai 'InfoDesk' lietotnē, bet arī uz sākuma ekrāna.	4.1.Veidot logrīkus nevis kā peldošos logus „InfoDesk" lietotnē, bet gan kā atsevišķas programmas ar atbilstošām klasēm.

Otrās iterācijas lietotājstāsts ir attēlots tabulā 3.2.2.

3.2.2.tabula

Otrā iterācija

Lietotāja stāsts	Uzdevums
3.Kā lietotājs, es vēlos, lai logrīkos vienmēr tiktu uzrādīta jaunākā informācija.	3.1.Pievienot visiem logrīkiem AlarmManager klasi, kas dos iespēju veikt periodiskus atjauninājumus.
	3.2.Pievienot AlarmManager klasei metodes, kas ļauj izsaukt un noņemt periodiskos atjauninājumus.
	3.3.Iestatīt automātisko atjauninājumu servisa sākšanos pie logrīka izveides.
5.Kā lietotājs, es vēlos, lai katram vienāda tipa logrīkam var uzstādīt atšķirīgus uzstādījumus.	5.1.Realizēt visu logrīku iestatījumu klases funkcionalitāti tā lai katram logrīga identifikācijas numuram būtu saglabāti atbilstošie uzstādījumi.
	5.2.Izveidot metodes, kas ļauj iegūt katra logrīka uzstādījumus un tos iestatīt konkrētajam logrīkam.
6.Kā lietotājs, es vēlos, lai 'JiraReaderWidget' logrīkam, katru dienu automātiski mainītos citāti.	6.1.Pārveidot „JiraReaderWidget" logrīka klases, kas atbild par datu nolasi tā, lai katru dienu no servera tiktu nolasīts

	atbilstošais citāts.
7.**Kā lietotājs, es vēlos, lai 'DateWidget' logrīku būtu iespējams attēlot ar mazajiem vai lielajiem burtiem.**	6.2.Pievienot papildus pogu konfigurācijas skata XML failā un attiecīgi realizēt tās funkcionalitāti.

Trešās iterācijas lietotājstāsts ir attēlots tabulā 3.2.3.

3.2.3.tabula

Trešā iterācija

Lietotāja stāsts	**Uzdevums**
9.Kā lietotājs, es vēlos, lai tiktu izveidots logrīks, kurš ļauj man uzlikt uz ekrāna ievadīto tekstu.	9.1.Izveidot logrīku pamata funkcionalitāti atbilstoši pārējo logrīku shēmai.
	9.2.Pievienot visus nepieciešamos logrīkas skatu XML failus.
	9.3.Pievienot logrīka konfigurācijas aktivitātei teksta lauku, kurā lietotājs ievadīs tekstu.
10.Kā lietotājs, es vēlos, lai ievadīto tekstu es jebkurā laikā varu nomainīt.	10.1.Izveidot izsaukumu, kurš lietotājam pieskaroties logrīka grafiskajai saskarnei atkārtoti izsauks logrīka konfigurācijas aktivitāti.
	10.2.Pievienot konfigurācijas aktivitātei trigeri, kurš saņemot izsaukumu atvērs konfigurācijas aktivitāti.

Ceturtās iterācijas lietotājstāsts ir attēlots tabulā 3.2.4.

3.2.4.tabula

Ceturtā iterācija

Lietotāja stāsts	**Uzdevums**
11.Kā lietotājs, es vēlos, lai	11.1.Izmainīt logrīka konfigurācijas

TextEditorWidget tekstam var nomainīt krāsu un izmēru.	aktivitātes XML failu, pievienojot, tam papildus funkcionalitātes pogas(teksta krāsas un burtu izmēra nomaiņai).
	11.2.Realizēt visu šo papildus pogu funkcionalitāti logrīka konfigurācijas aktivitātē.
12.Kā lietotājs, es vēlos, lai TekstEditorWidget tekstu var rakstīt slīprakstā, treknrakstā, ar pasvītrojumu, vai ar pārsvītrojumu.	12.1.Izmainīt logrīka konfigurācijas aktivitātes XML failu, pievienojot, tam papildus funkcionalitātes pogas.
	12.2.Realizēt visu šo papildus pogu funkcionalitāti logrīka konfigurācijas aktivitātē.
13.Kā lietotājs, es vēlos, lai TekstEditorWidget logrīks ļauj atsevišķi apstrādāt katru teksta simbolu.	13.1.Pievienot logrīkam „TextWatcher" funkciju, kas veiks katra atsevišķā simbola apstrādi.

Piektās iterācijas lietotājstāsts ir attēlots tabulā 3.2.5.

3.2.5.tabula

Piektā iterācija

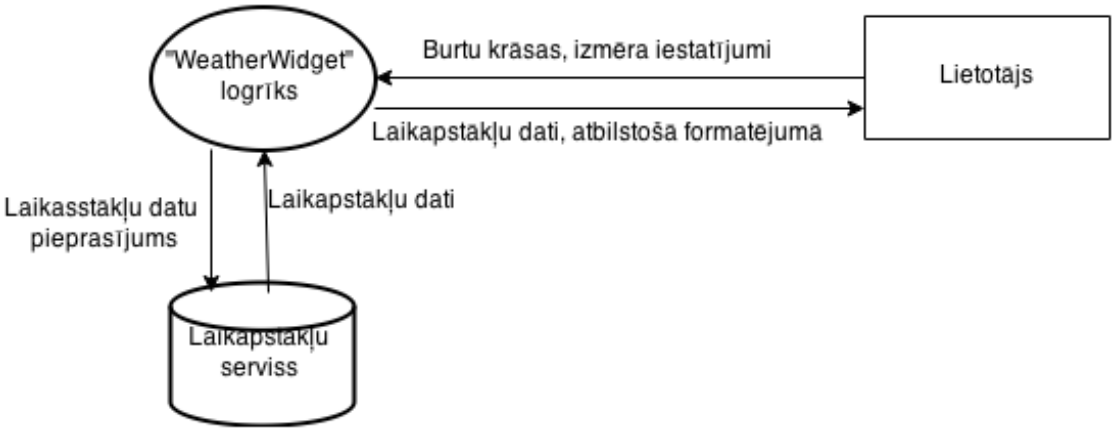
Lietotāja stāsts	Uzdevums
8.Kā lietotājs, es vēlos, lai uzstādītās logrīku knfigurācijas varētu pārnest uz citu ierīci.	8.1.Izveidot klasi „InfoDesk" lietotnē, kuras uzdevums būs noteikt cik daudz logrīku pašlaik atrodas lietotnē.
	8.2.Izveidot metode jau esošā „InfoDesk klasē", kuras uzdevums būs izsaukt katra logrīka konfigurāciju XML failu pārvietošanu uz GoogleDrive disku.

4. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

Programmatuūras projektējuma apraksts, ir paredzēts, lai aprakstītu kā lietotājstāstu nodaļā minētās prasības tika realizētas galaprodukta izstrādē. Šī nodaļa tika izveidota laika gaitā, katru reizi pēc jaunu lietotājstāstu iegūšanas un to precīzākas nodefinēšanas, projektējuma apraksts tika papildināts.

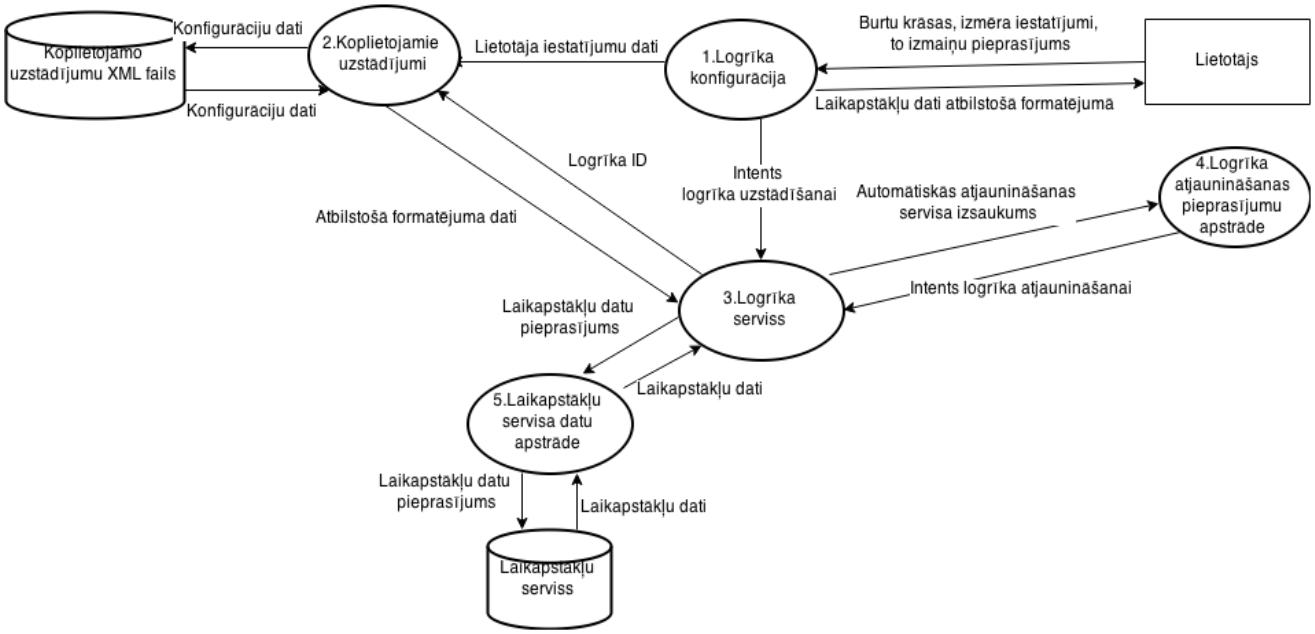
4.1. Programmatūras datu plūsmu diagrammas

„WeatherWidget" logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.1.



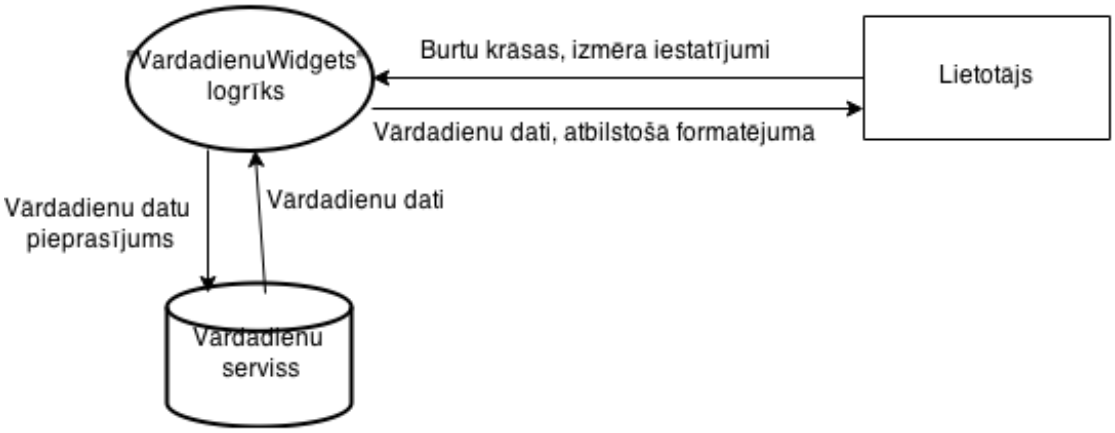
4.1.1. att. **"WeatherWidget" logrīka 0. līmeņa datu plūsmu diagramma**

„WeatherWidget" logrīka 1. līmeņa datu plūsmu diagramma redzama attēlā 4.1.2



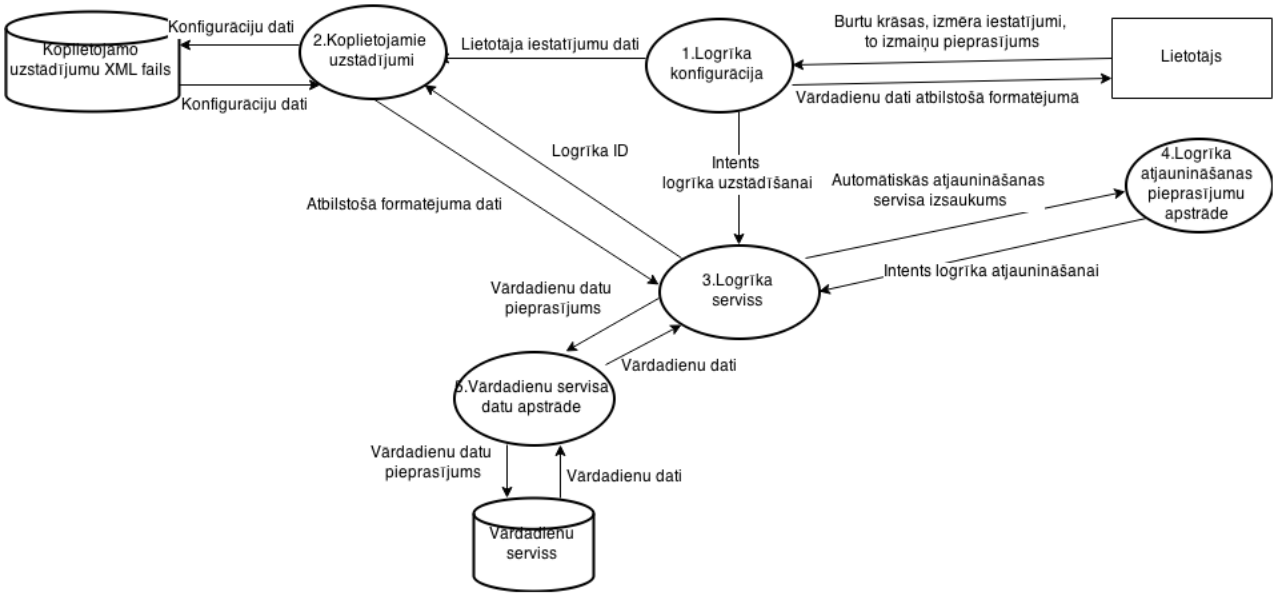
.4.1.2. att. **"WeatherWidget" logrīka 1. līmeņa datu plūsmu diagramma**

„VardadienuWidget” logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.3.



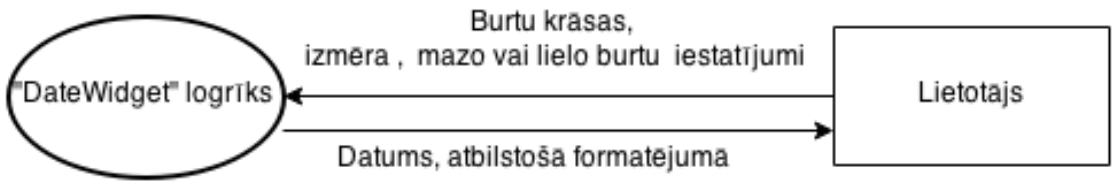
4.1.4. att. **”VardadienuWidget” logrīka 0. līmeņa datu plūsmu diagramma**

„VardadienuWidget” logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.3.



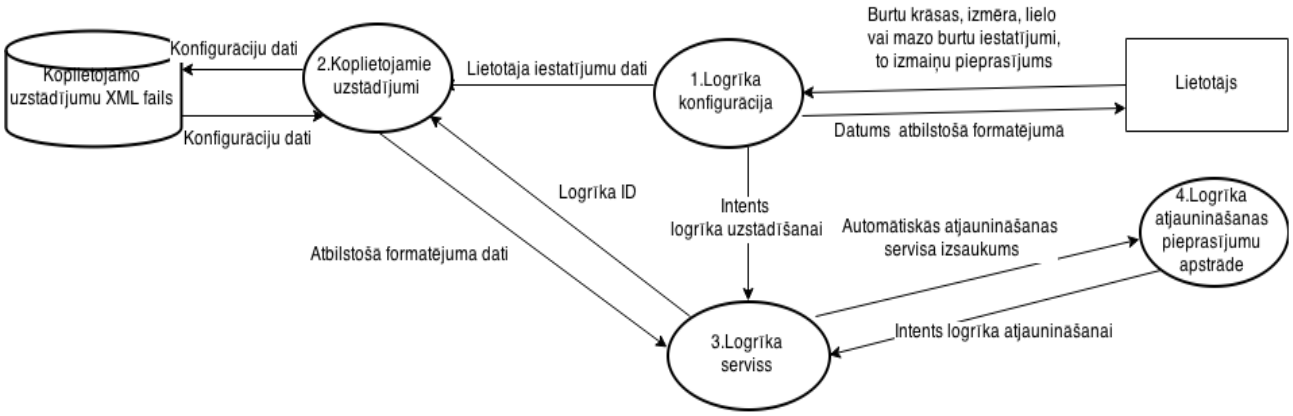
4.1.4. att. **”VardadienuWidget” logrīka 1. līmeņa datu plūsmu diagramma**

„DateWidget” logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.5.



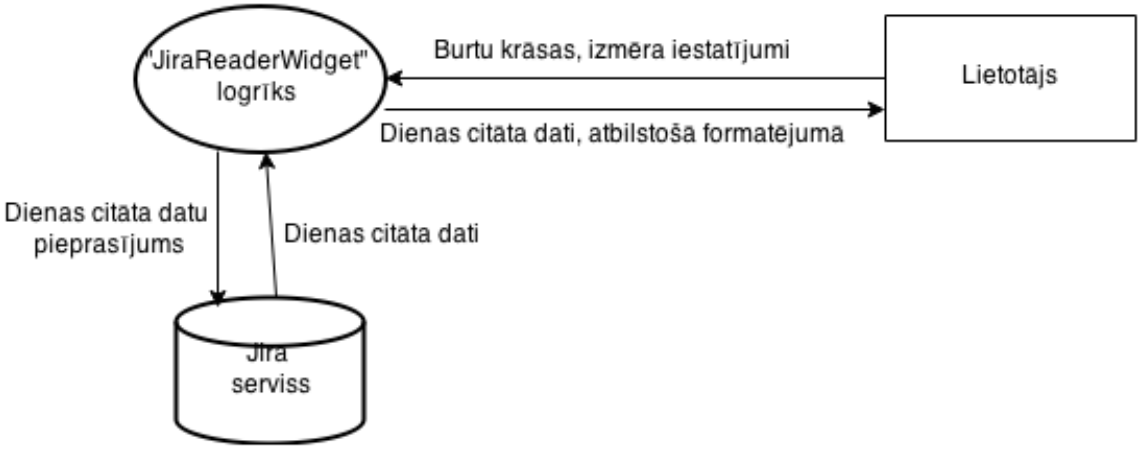
4.1.5. att. **”DateWidget” logrīka 0. līmeņa datu plūsmu diagramma**

„DateWidget” logrīka 1. līmeņa datu plūsmu diagramma redzama attēlā 4.1.6.



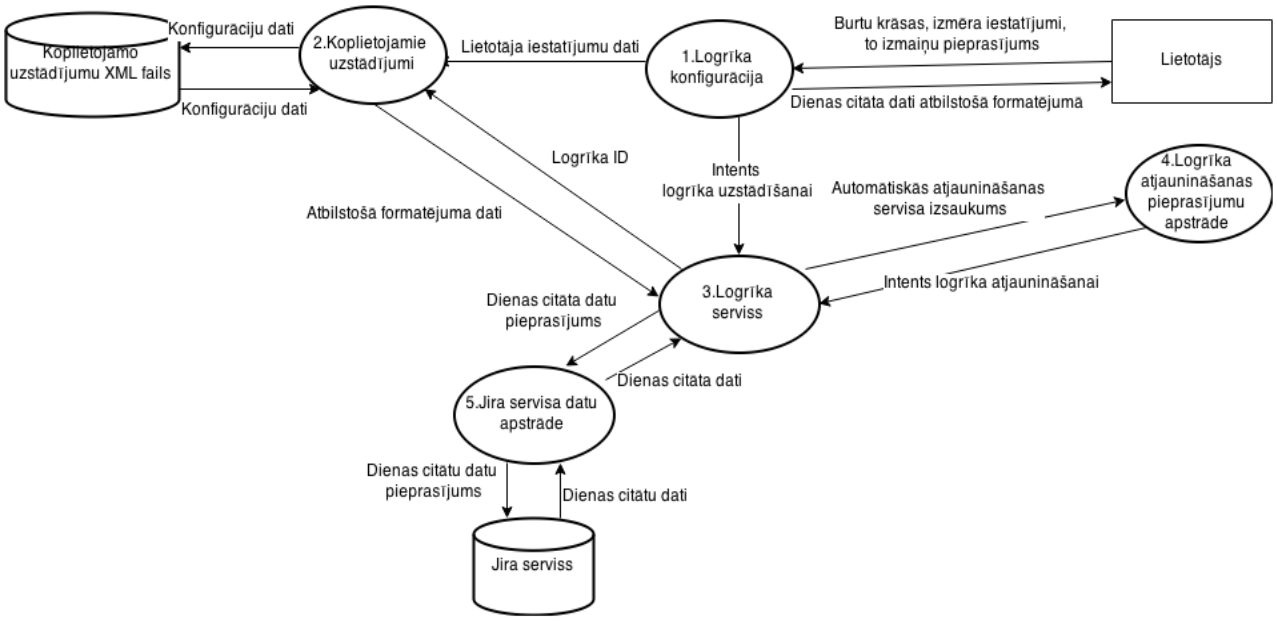
4.1.6. att. **”DateWidget” logrīka 1. līmeņa datu plūsmu diagramma**

„JiraReaderWidget” logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.7.



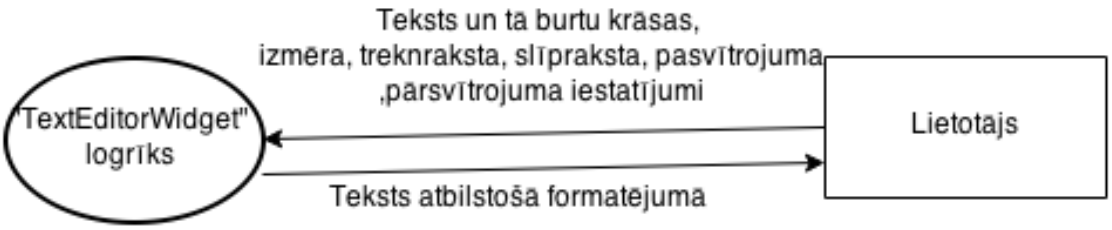
4.1.7. att. **”JiraReaderWidget” logrīka 0. līmeņa datu plūsmu diagramma**

„JiraReaderWidget” logrīka 1. līmeņa datu plūsmu diagramma redzama attēlā 4.1.8.



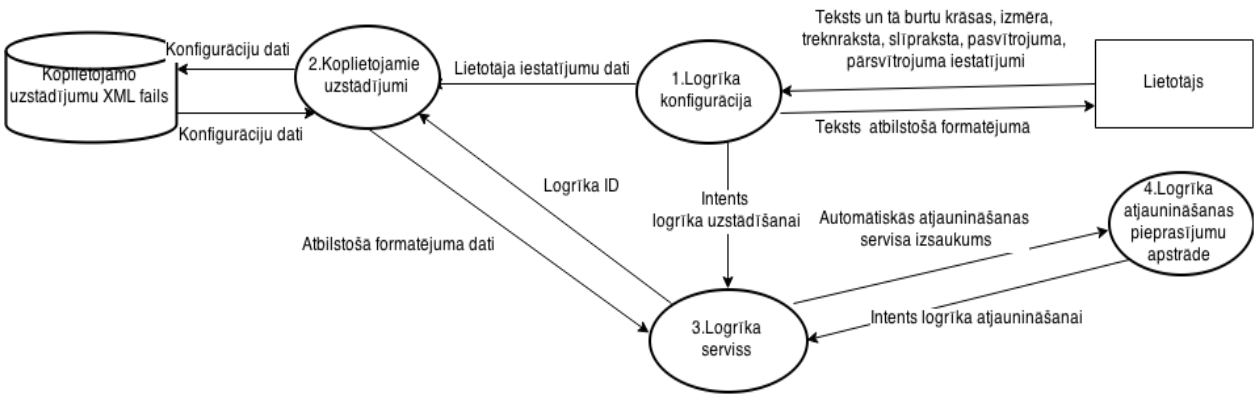
4.1.8. att. **”JiraReaderWidget” logrīka 1. līmeņa datu plūsmu diagramma**

„TextEditorWidget” logrīka 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.9.



4.1.9. att. **”TextEditorWidget” logrīka 0. līmeņa datu plūsmu diagramma**

„TextEditorWidget” logrīka 1. līmeņa datu plūsmu diagramma redzama attēlā 4.1.10.



4.1.10. att. **”TextEditorWidget” logrīka 1. līmeņa datu plūsmu diagramma**

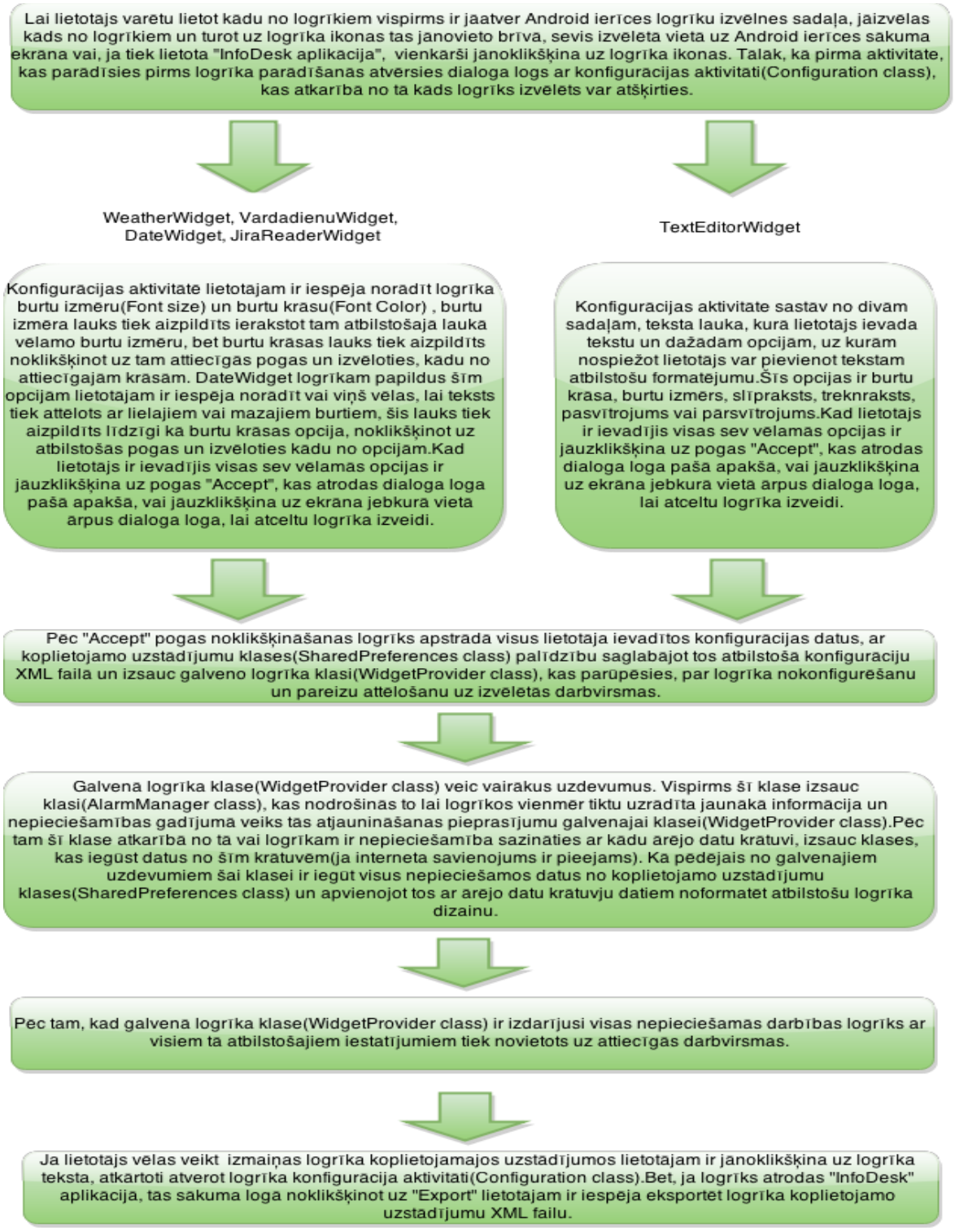
„InfoDesk” lietotnes 0. līmeņa datu plūsmu diagramma redzama attēlā 4.1.11.

4.1.11. att. **”InfoDesk” lietotnes 0. līmeņa datu plūsmu diagramma**

Datu plūsmu diagrammas tika izveidotas pēc visu lietotājstāstu prasību iegūšanas. To veidošanai tika izmantots draw.io (6. avots) rīks.

4.2. Funkcionalitātes piemērs

Lai labāk izprastu visu logrīku darbība, tika izveidots funkcionalitātes piemērs (*4.2.1. attēls*), tā kā visu logrīku darbības principi ir līdzīgs, šis piemērs tika veidots, kā vispārējs paraugs tam kā darboties ar jebkuru no logrīkiem.

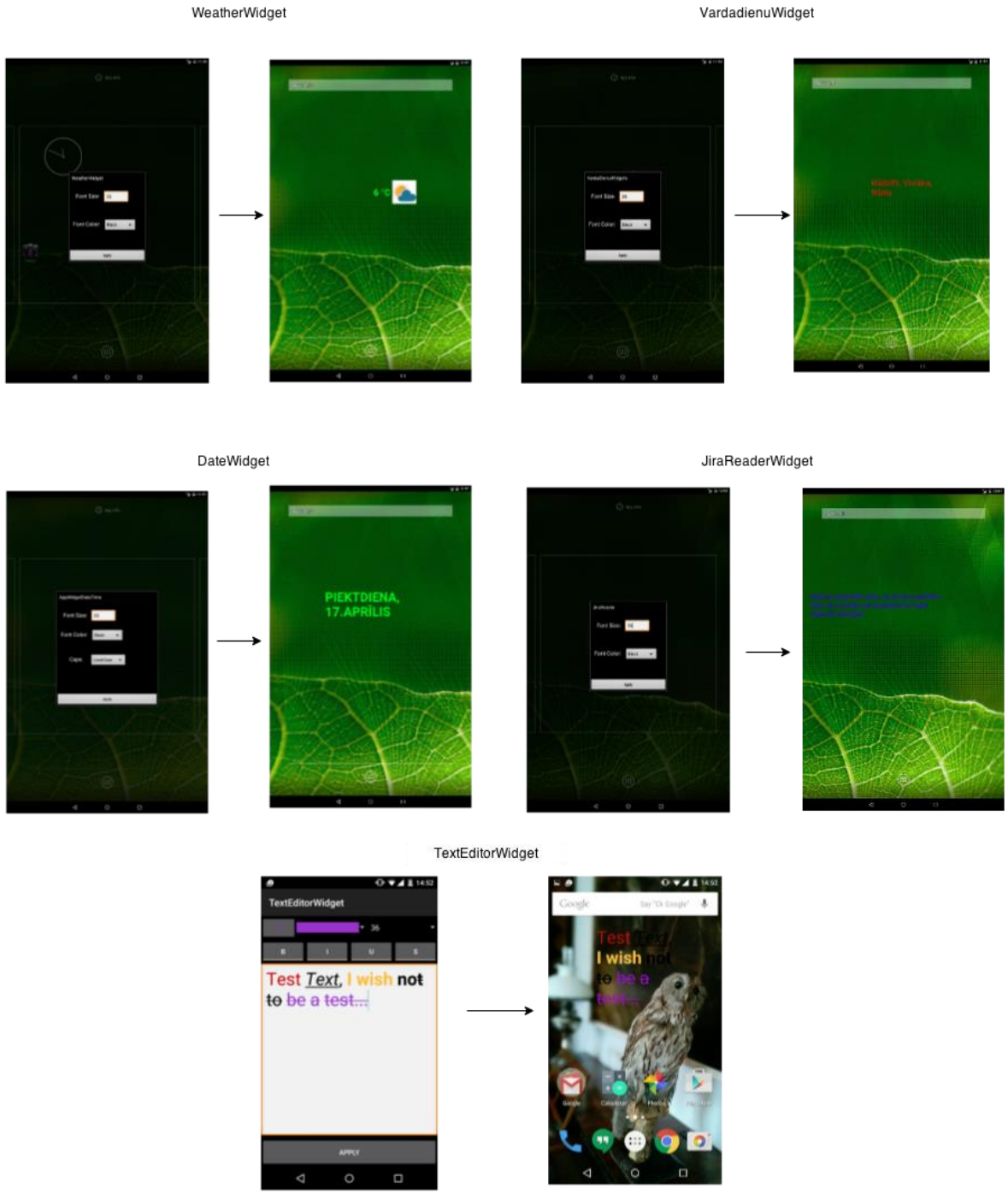


4.2.1. att. Funkcionalitātes piemērs pasūtījuma veikšanai

4.3. Programmatūras saskarņu diagrammas

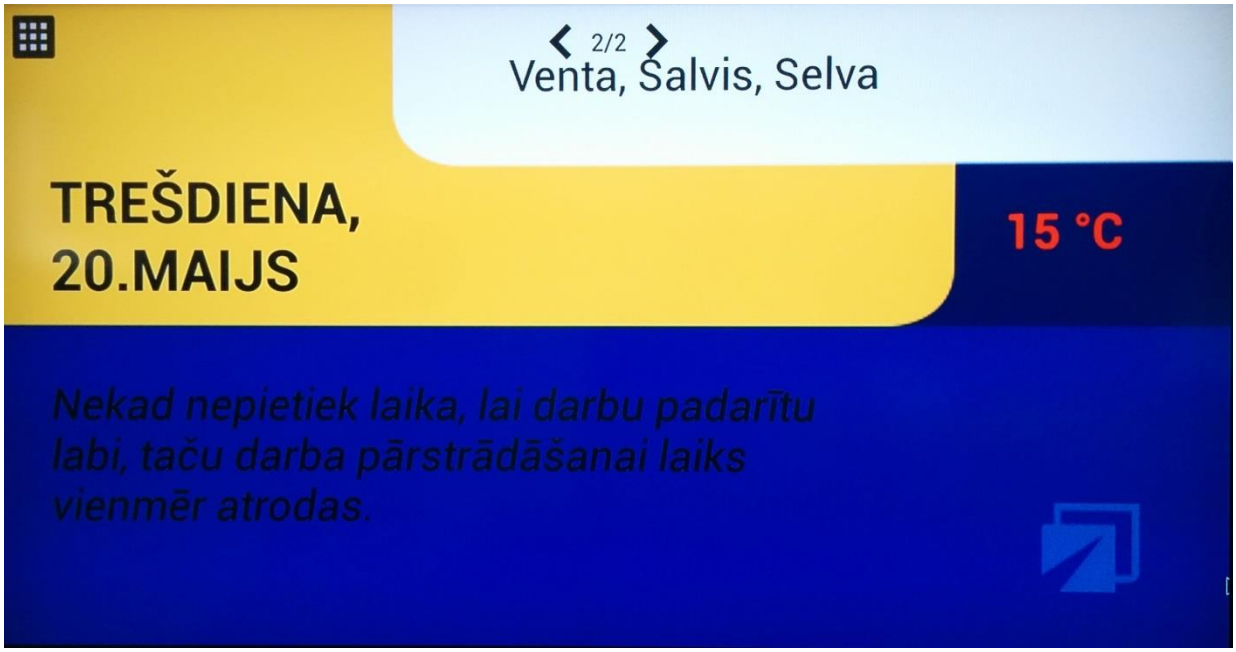
Diagrammās ir attēlots, kā izskatās katra logrīka lietotāju saskarnes.

Visu logrīku saskarņu diagrammas redzamas attēlā 4.3.1.



4.3.1. att. **Visu logrīku saskarņu diagrammas**

Tas kā logrīki izskatās, kad tie ir novietoti „Infodesk" lietotnē redzams attēlā 4.3.2.



***4.3.2. att.* Logrīki „InfoDesk" lietotnē**

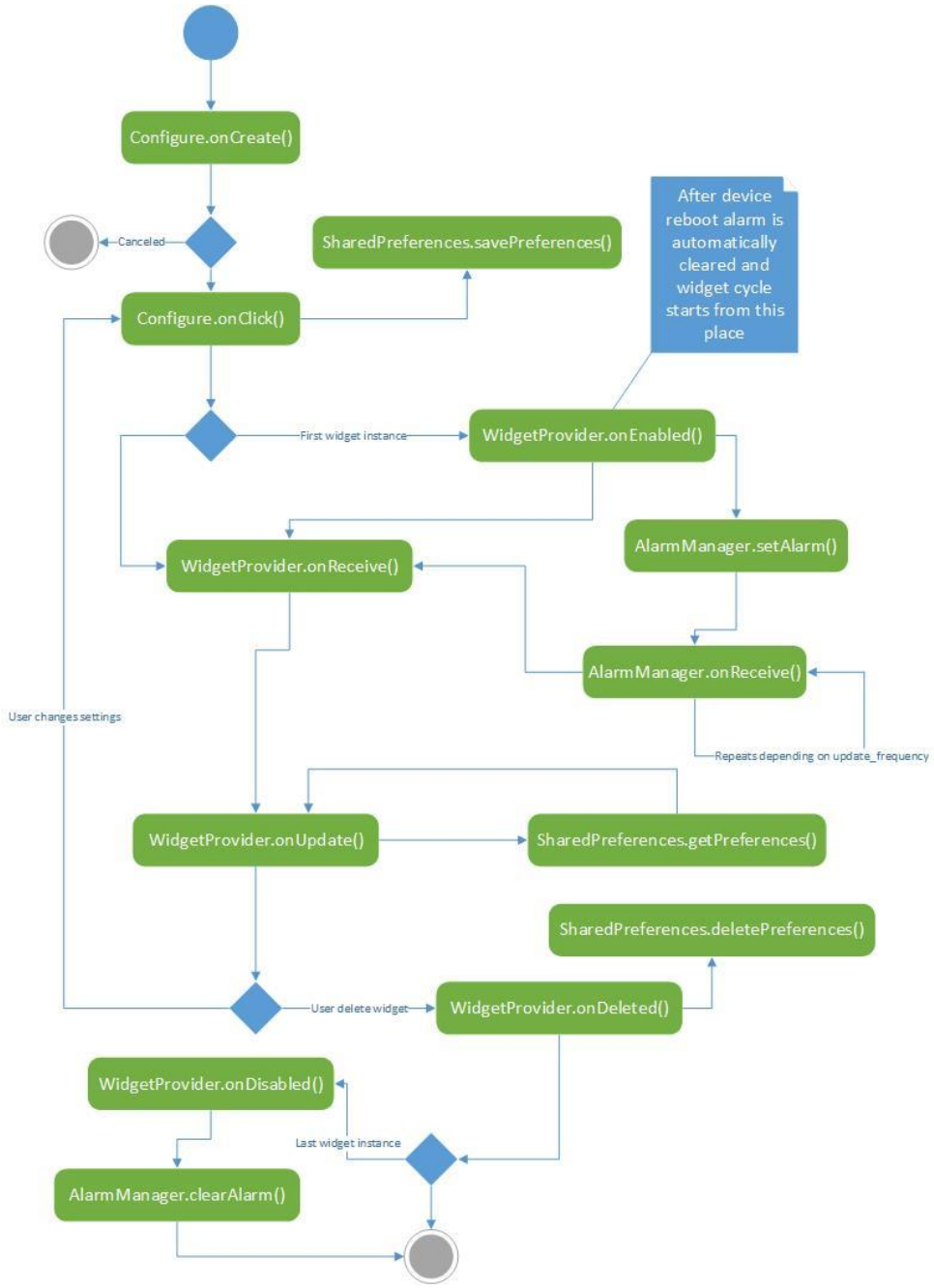
„InfoDesk" logrīku eksportēšana funkcijas grafiskā lietotāja saskarne redzama attēlā 4.3.3.

4.3.3. att. Logrīku eksportēšanas funkcijas grafiskā saskarne lietotnē „InfoDesk"

4.4. UML diagrammas

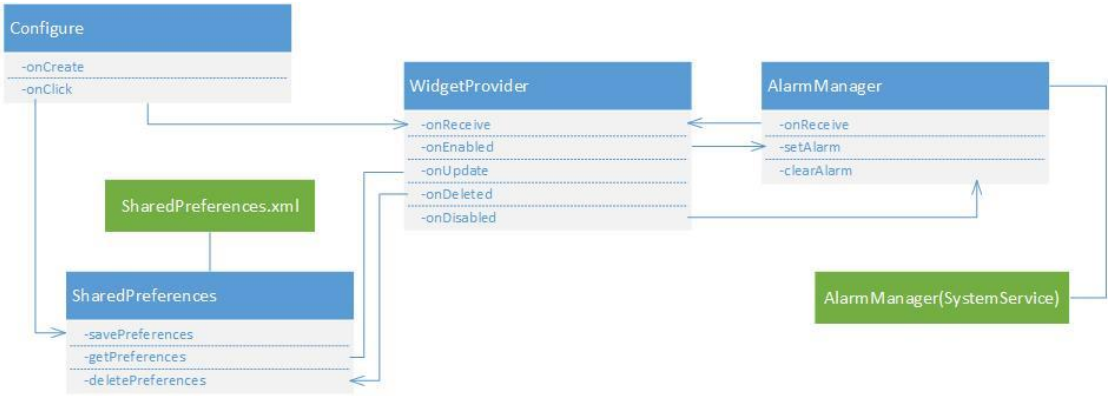
Lai izstrādātājs labāk izprastu to kā logrīka klases un to metodes ir saistītas savā starpā, tika izstrādātas 2 UML diagrammas.Šajās diagrammās ir parādīta tikai visu logrīku pamata funkcionalitāte, dziļāk neapskatos katram logrīkam specializēto funkcionalitāti un klases.Diagrammu veidošanai tika izmantota lietotne Micosoft Visio 2013(8.avots).

Logrīku UML aktivitāšu diagramma parādīta attēlā 4.4.1.



4.4.1. att. Logrīku UML aktivitāšu diagramma

Logrīku UML klašu diagramma parādīta attēlā 4.4.2.



4.4.2. att. Logrīku UML klašu diagramma

4.5.WeatherWidget moduļu projektējums

4.5.1. Logrīka konfigurācijas moduļa projektējums

4.5.1.1. Klase Configure

Klase izveido konfigurācijas skatu, kurš dod iespēju lietotājam norādīt kādu logrīka burtu izmēru un krāsu viņš vēlas, pēc šo datu ievades klase saglabā šos uzstādījumus izmantojot klases WeatherWidgetSharedPreferences metodes un izsauc WidgetProvider klasi, kas veic tālāku šo datu apstrādi un nepieciešamo papildklašu izsaukšanu.

Metodes:

Tips	Metode	Apraksts
void	OnCreate(savedInstanceState)	Ielādē konfigurācijas aktivitātes skatu un inicializē mainīgo widgetID.
void	onClick(View v)	Veic lietotāja konfigurāciju datu priekšapstrādi.

Mainīgie:

Tips	Mainīgais	Apraksts
Bundle	extras	Logrīka izveides brīdī izveidoto datu kopa.
Configure	context	Konfigurācijas aktivitātes konteksts.
int	widgetID	Satur tikko izveidotā logrīka ID.
Spinner	fontColorButton	Norāde uz burtu krāsas izvēles pogas skatu.
EditText	fontSizeButton	Norāde uz burtu izmēra ievades lauka skatu.
Button	acceptButton	Norāde uz ‘accept’ pogas

		skatu.
int	fontSize	Burtu izmērs.
String	fontColor	Burtu krāsa.
RemoteViews	views	Norāde uz galveno logrīka skatu.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
Intent	resultValue	Mainīgais, kurš glabā norādi uz WidgetProvider klasi.

4.5.2. Logrīka koplietojamo uzstādījumu moduļa projektējums

4.5.2.1. Klase WeatherWidgetSharedPreferences

Šī klase satur metodes, kas veic lietotāja ievadīto konfigurācijas datu saglabāšanu, nolasīšanu un dzēšanu no logrīka koplietojamo uzstādījumu XML formāta faila.

Metodes:

Tips	**Metode**	**Apraksts**
void	savePreferences(int FontSize, String FontColor,int widgetID, Context context)	Fukcija, kas saglabā lietotāja norādītos iestatījumus.
int	getFontSize(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu izmēru.
String	getFontColor(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu krāsu.
void	saveDate(String date, Context context)	Funkcijas, kas saglabā pašreizējo ierīces laiku.
String	String getDate(Context context)	Funkcijas, kas atgriež iepriekšējo logrīka

		atjaunināšanas laiku.
void	deletePreferences(int widgetID, Context context)	Funkcija, kas izdzēš logrīka koplietojamo uzstādījumu ierakstus no to XML faila.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
Editor	editor	Mainīgais, kas satur metodes, kas ļauj veikt izmaiņas koplietojamo uzstādījumu XML failā.
int	fontSize	Logrīka burtu izmērs.
String	fontColor	Logrīka burtu krāsa.
String	date	Logrīka pēdējā atjaunināšanas brīža laiks.

4.5.3. Logrīka servisa moduļa projektējums

4.5.3.1. Klase WidgetProvider

Galvenā logrīka klase, kas atbildīga par logrīka datu pareizu attēlošanu tā galvenajā skatā un to periodisku atjaunināšanu. Šī klase nepieciešamības gadījumā var izsaukt visas pārējās logrīka klases.

Metodes:

Tips	**Metode**	**Apraksts**
boolean	netConnect(Context ctx)	Funkcija, kas pārbauda vai Android ierīcei ir pieejams interneta savienojums.
void	onDeleted(Context context, int[] appWidgetIds)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie konkrēta logrīka izdzēšanas.

void	onDisabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie visu logrīka izdzēšanas. Izsauc, WeatherWidgetAlarmManager klases metodi clearAlarms, kas pārtrauc automātisko logrīku atjauninšanu.
void	onEnabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie pirmā logrīka novietošanas uz ierīces ekrāna vai ierīces pārstartēšanas. Izsauc, WeatherWidgetAlarmManager klases metodi setAlarms, kas nodrošina automātisko logrīku atjauninšanu.
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninašanas pieprasījumus un izsauc onUpdate funkciju.
void	onUpdate(Context context, AppWidgetManager, appWidgetManager, int[] appWidgetIds)	Nolasa visu logrīku ID un katram no tiem izsauc funkciju widgetUpdate, kas veic logrīku atjauninašanu.
void	widgetUpdate(Context context,int widgetID)	Funkcija, kas veic logrīku atjauninašanu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
AppWidgetManager	appWidgetManager	Satur informāciju par visiem logrīkiem.
boolean	reboot	Mainīgais, kurš glabā informāciju par to vai

		ierīcei pirms pēdējās atjaunināšans ir bijusi pārstartēšana.
DateFormat	df	Satur informāciju, par to kādā formātā ir jāformatē datums.
HandleJSON	obj	HandleJSON klases objekts ar kura palīdzību logrīks iegūst informāciju no ārējajiem servisiem.
int	count	Satur konkrētā tipa logrīku skaitu.
int	fontSize	Burtu izmērs.
int[]	ids	Masīvs ar visiem konkrētā tipa logrīku ID numuriem.
RemoteViews	views	Norāde uz galveno logrīka skatu.
String	date	Ierīces laiks.
String	fontColor	Burtu krāsa.
String	lastDate	Pēdējā logrīka atjaunināšanas brīža laiks.
String	UPDATE_WIDGETS	Logrīka atjauninšanas pieprasījuma teksts.
String	url	Tīmekļa adrese, no kuras tiek iegūta informācija par laikapstākļiem.
String	weather	Laikapstākļu dati.
NetworkInfo	info	Informācija par to vai ierīcei pieejams Interneta pieslēgums.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā

		izsaukšanas.
ConnectivityManager	cm	Mainīgais, kurš satur informāciju par ierīces tīkla pieslēgumu.

4.5.4. Logrīka atjauninašnas pieprasījuma apstrādes moduļa projektējums

4.5.4.1. Klase WeatherWidgetAlarmManager

Klase, kas nodrošina periodisku logrīka datu atjaunināšanu, pēc klasē noteiktā laika, nosūtot atjaunināšanas pieprasījumu galvenajai logrīka klasei WidgetProvider.

Metodes:

Tips	Metode	Apraksts
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninājumu pieprasījumu.
void	setAlarm(Context context)	Funkcija, kas uzstāda logrīku automātisko atjaunošanos pēc noteikta laika intervāla.
void	clearAlarm(Context context)	Funkcija, kas noņem logrīku automātisko atjaunināšanos.

Mainīgie:

Tips	Mainīgais	Apraksts
String	ALARM_ACTION	Logrīka atjaunināšanas pieprasījuma teksts.
long	UPDATE_FREQUENCY	Logrīka atjaunināšanas biežums.
String	action	Satur informāciju par saņemto pieprasījumu.

Intent	updateIntent	Satur norādi uz WeatherWidgetAlarmManager klasi.
AlarmManager	alarmManager	Sistēmas servisa objekts.
PendingIntent	pIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.5.5. Logrīka laikapstākļu servisa datu apstrādes moduļa projektējums

4.5.4.1. Klase HandleJSON

Klase, kas pēc pieprasījuma no galvenās logrīka klases WidgetProvider veic datu nolasi un apstrādi no logrīka servisa JSON faila.

Metodes:

Tips	**Metode**	**Apraksts**
konstruktors	HandleJSON(String url)	Inicializē mainīgo urlString.
String	getTemperature()	Funkcija, kas atgriež temperatūras datus.
String	getDescription()	Funkcija, kas atgriež laikapstākļu apraksta datus.
void	readAndParseJSON(String in)	Nolasa nepieciešamos datus no padotajiem datiem.
void	fetchJSON()	Savienojas ar datu serveri un apstrādā datu

		ieguves procesu.
void	run()	Nodrošina funkcijas fetchJSon() veiksmīgu izpildi.
String	convertStreamToString(java.io.InputStream is)	Konvertē padoto datu plūsmu uz string datu tipu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
String	description	Satur laikapstākļu aprakstu.
String	temperature	Satur gaisa temperatūru.
String	urlString	Satur tīmekļa vietnes adresi no kuras tiks ielādēti laikapstākļu dati.
boolean	parsingComplete	Mainīgais, kurš satur informāciju par to vai datu apstrāde ir veikta.
JSONObject	reader	Objekts, kurā tiek ievadīta ievades datu plūsma pirms tās apstrādes.
JSONObject	main	Satur konkrēto datni no visas datu plūsmas.
InputStream	stream	Satur ievades datu plūsmu.
String	data	No servisa iegūtie dati pēc to apstrādes.
JSONArray	array	Masīvs ar

		nepieciešamo laikapstākļu informāciju.
URL	url	Objekts, kurš satur datu ielādes servera tīmekļa adresi.
HttpURLConnection	conn	Objekts, kurš nodrošina savienojumu ar datu serveri.
java.util.Scanner	scanner	Ievades plūsmas apstrādes objekts.

4.6.VardaDienuWidget moduļu projektējums

4.6.1. Logrīka konfigurācijas moduļa projektējums

4.6.1.1. Klase Configure

Klase izveido konfigurācijas skatu, kurš dod iespēju lietotājam norādīt kādu logrīka burtu izmēru un krāsu viņš vēlas, pēc šo datu ievades klase saglabā šos uzstādījumus izmantojot klases NameWidgetSharedPreferences metodes un izsauc WidgetProvider klasi, kas veic tālāku šo datu apstrādi un nepieciešamo papildklašu izsaukšanu.

Metodes:

Tips	Metode	Apraksts
void	OnCreate(savedInstanceState)	Ielādē konfigurācijas aktivitātes skatu un inicializē mainīgo widgetID.
void	onClick(View v)	Veic lietotāja konfigurāciju datu priekšapstrādi.

Mainīgie:

Tips	Mainīgais	Apraksts
Bundle	extras	Logrīka izveides brīdī izveidoto datu kopa.
Configure	context	Konfigurācijas aktivitātes konteksts.
int	widgetID	Satur tikko izveidotā logrīka ID.
Spinner	fontColorButton	Norāde uz burtu krāsas izvēles pogas skatu.
EditText	fontSizeButton	Norāde uz burtu izmēra ievades lauka skatu.
Button	acceptButton	Norāde uz ‘accept’ pogas skatu.
int	fontSize	Burtu izmērs.
String	fontColor	Burtu krāsa.
RemoteViews	views	Norāde uz galveno logrīka skatu.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
Intent	resultValue	Mainīgais, kurš glabā norādi uz WidgetProvider klasi.

4.6.2. Logrīka koplietojamo uzstādījumu moduļa projektējums

4.6.2.1. Klase NameWidgetSharedPreferences

Šī klase satur metodes, kas veic lietotāja ievadīto konfigurācijas datu saglabāšanu, nolasīšanu un dzēšanu no logrīka koplietojamo uzstādījumu XML formāta faila.

Metodes:

Tips	Metode	Apraksts
void	savePreferences(int FontSize, String FontColor,int widgetID, Context context)	Fukcija, kas saglabā lietotāja norādītos iestatījumus.
int	getFontSize(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu izmēru.
String	getFontColor(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu krāsu.
void	deletePreferences(int widgetID, Context context)	Funkcija, kas izdzēš logrīka koplietojamo uzstādījumu ierakstus no to XML faila.

Mainīgie:

Tips	Mainīgais	Apraksts
Editor	editor	Mainīgais, kas satur metodes, kas ļauj veikt izmaiņas koplietojamo uzstādījumu XML failā.
int	fontSize	Logrīka burtu izmērs.
String	fontColor	Logrīka burtu krāsa.

4.6.3. Logrīka servisa moduļa projektējums

4.6.3.1. Klase WidgetProvider

Galvenā logrīka klase, kas atbildīga par logrīka datu pareizu attēlošanu tā galvenajā skatā un to periodisku atjaunināšanu. Šī klase nepieciešamības gadījumā var izsaukt visas pārējās logrīka klases.

Metodes:

Tips	Metode	Apraksts
boolean	netConnect(Context ctx)	Funkcija, kas pārbauda vai Android ierīcei ir pieejams interneta savienojums.
void	onDeleted(Context context, int[] appWidgetIds)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie konkrēta logrīka izdzēšanas.
void	onDisabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie visu logrīka izdzēšanas. Izsauc, NameWidgetAlarmManager klases metodi clearAlarms, kas pārtrauc automātisko logrīku atjauninšanu.
void	onEnabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie pirmā logrīka novietošanas uz ierīces ekrāna vai ierīces pārstartēšanas. Izsauc, NameWidgetAlarmManager klases metodi setAlarms, kas nodrošina automātisko logrīku atjauninšanu.
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjaunināšanas pieprasījumus un izsauc onUpdate funkciju.
void	onUpdate(Context context,	Nolasa visu logrīku ID un

	AppWidgetManager, appWidgetManager, int[] appWidgetIds)	katram no tiem izsauc funkciju widgetUpdate, kas veic logrīku atjaunināšanu.
void	widgetUpdate(Context context,int widgetID)	Funkcija, kas veic logrīku atjaunināšanu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
AppWidgetManager	appWidgetManager	Satur informāciju par visiem logrīkiem.
HandleJSON	obj	HandleJSON klases objekts ar kura palīdzību logrīks iegūst informāciju no ārējajiem servisiem.
int	count	Satur konkrētā tipa logrīku skaitu.
int	fontSize	Burtu izmērs.
int[]	ids	Masīvs ar visiem konkrētā tipa logrīku ID numuriem.
RemoteViews	views	Norāde uz galveno logrīka skatu.
String	fontColor	Burtu krāsa.
String	UPDATE_WIDGETS	Logrīka atjauninšanas pieprasījuma teksts.
String	url	Tīmekļa adrese, no kuras tiek iegūta informācija par laikapstākļiem.
NetworkInfo	info	Informācija par to vai ierīcei pieejams Interneta pieslēgums.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.

PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
ConnectivityManager	cm	Mainīgais, kurš satur informāciju par ierīces tīkla pieslēgumu.

4.6.4. Logrīka atjaunināšnas pieprasījuma apstrādes moduļa projektējums

4.6.4.1. Klase NameWidgetAlarmManager

Klase, kas nodrošina periodisku logrīka datu atjaunināšanu, pēc klasē noteiktā laika, nosūtot atjaunināšanas pieprasījumu galvenajai logrīka klasei WidgetProvider.

Metodes:

Tips	**Metode**	**Apraksts**
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninājumu pieprasījumu.
void	setAlarm(Context context)	Funkcija, kas uzstāda logrīku automātisko atjaunošanos pēc noteikta laika intervāla.
void	clearAlarm(Context context)	Funkcija, kas noņem logrīku automātisko atjaunināšanos.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
String	ALARM_ACTION	Logrīka atjaunināšanas pieprasījuma teksts.
long	UPDATE_FREQUENCY	Logrīka atjaunināšanas biežums.

String	action	Satur informāciju par saņemto pieprasījumu.
Intent	updateIntent	Satur norādi uz NameWidgetAlarmManager klasi.
AlarmManager	alarmManager	Sistēmas servisa objekts.
PendingIntent	pIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.6.5. Logrīka vārdadienu servisa datu apstrādes moduļa projektējums

4.6.5.1. Klase HandleJSON

Klase, kas pēc pieprasījuma no galvenās logrīka klases WidgetProvider veic datu nolasi un apstrādi no logrīka servisa JSON faila.

Metodes:

Tips	Metode	Apraksts
konstruktors	HandleJSON(String url)	Inicializē mainīgo urlString.
String	getNames()	Funkcija, kas atgriež vārdadienu datus.
String	getDescription()	Funkcija, kas atgriež laikapstākļu apraksta datus.
void	readAndParseJSON(String in)	Nolasa nepieciešamos datus no padotajiem datiem.
void	fetchJSON()	Savienojas ar datu serveri un

		apstrādā datu ieguves procesu.
void	run()	Nodrošina funkcijas fetchJSon() veiksmīgu izpildi.
String	convertStreamToString(java.io.InputStream is)	Konvertē padoto datu plūsmu uz string datu tipu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
String	messages	Mainīgais, kurā glabās vārdus.
String	urlString	Satur tīmekļa vietnes adresi no kuras tiks ielādēti vārdadienu dati.
int	m	Mēneša kārtas numurs.
int	d	Mēneša dienas kārtas numurs.
boolean	parsingComplete	Mainīgais, kurš satur informāciju par to vai datu apstrāde ir veikta.
InputStream	stream	Satur ievades datu plūsmu.
String	data	No servisa iegūtie dati pēc to apstrādes.
URL	url	Objekts, kurš satur datu ielādes servera tīmekļa adresi.
HttpURLConnection	conn	Objekts, kurš nodrošina savienojumu

		ar datu serveri.
String	sep	Vārdu atdalītājs.
java.util.Scanner	scanner	Ievades plūsmas apstrādes objekts.

4.7.DateWidget moduļu projektējums

4.7.1. Logrīka konfigurācijas moduļa projektējums

4.7.1.1. Klase Configure

Klase izveido konfigurācijas skatu, kurš dod iespēju lietotājam norādīt vai viņš vēlas logrīka tekstu ar mazajiem burtiem, vai lielajiem un kādu logrīka burtu izmēru un krāsu viņš vēlas, pēc šo datu ievades klase saglabā šos uzstādījumus izmantojot klases DateWidgetSharedPreferences metodes un izsauc WidgetProvider klasi, kas veic tālāku šo datu apstrādi un nepieciešamo papildklašu izsaukšanu.

Metodes:

Tips	Metode	Apraksts
void	OnCreate(savedInstanceState)	Ielādē konfigurācijas aktivitātes skatu un inicializē mainīgo widgetID.
void	onClick(View v)	Veic lietotāja konfigurāciju datu priekšapstrādi.

Mainīgie:

Tips	Mainīgais	Apraksts
Bundle	extras	Logrīka izveides brīdī izveidoto datu kopa.
Configure	context	Konfigurācijas aktivitātes konteksts.
int	widgetID	Satur tikko izveidotā logrīka ID.

Spinner	fontColorButton	Norāde uz burtu krāsas izvēles pogas skatu.
EditText	fontSizeButton	Norāde uz burtu izmēra ievades lauka skatu.
Button	acceptButton	Norāde uz ‘accept’ pogas skatu.
int	fontSize	Burtu izmērs.
String	fontColor	Burtu krāsa.
RemoteViews	views	Norāde uz galveno logrīka skatu.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
Spinner	capsButton	Mazie burti vai lielie burti.
Intent	resultValue	Mainīgais, kurš glabā norādi uz WidgetProvider klasi.

4.7.2. Logrīka koplietojamo uzstādījumu moduļa projektējums

4.7.2.1. Klase DateWidgetSharedPreferences

Šī klase satur metodes, kas veic lietotāja ievadīto konfigurācijas datu saglabāšanu, nolasīšanu un dzēšanu no logrīka koplietojamo uzstādījumu XML formāta faila.

Metodes:

Tips	**Metode**	**Apraksts**
void	savePreferences(int FontSize, String FontColor, String Caps, int widgetID, Context context)	Fukcija, kas saglabā lietotāja norādītos iestatījumus.
int	getFontSize(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu izmēru.

String	getFontColor(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu krāsu.
void	deletePreferences(int widgetID, Context context)	Funkcija, kas izdzēš logrīka koplietojamo uzstādījumu ierakstus no to XML faila.
String	getCaps(int widgetID, Context context)	Funkcija, kas atgriež to vai logrīkam nepieciešami lielie vai mazie burti.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
Editor	editor	Mainīgais, kas satur metodes, kas ļauj veikt izmaiņas koplietojamo uzstādījumu XML failā.
int	fontSize	Logrīka burtu izmērs.
String	fontColor	Logrīka burtu krāsa.
String	caps	Lielie vai mazie burti.

4.7.3. Logrīka servisa moduļa projektējums

4.7.3.1. Klase WidgetProvider

Galvenā logrīka klase, kas atbildīga par logrīka datu pareizu attēlošanu tā galvenajā skatā un to periodisku atjaunināšanu. Šī klase nepieciešamības gadījumā var izsaukt visas pārējās logrīka klases.

Metodes:

Tips	**Metode**	**Apraksts**
void	onDeleted(Context context, int[] appWidgetIds)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie

		konkrēta logrīka izdzēšanas.
void	onDisabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie visu logrīka izdzēšanas. Izsauc, DateWidgetAlarmManager klases metodi clearAlarms, kas pārtrauc automātisko logrīku atjauninšanu.
void	onEnabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie pirmā logrīka novietošanas uz ierīces ekrāna vai ierīces pārstartēšanas. Izsauc, DateWidgetAlarmManager klases metodi setAlarms, kas nodrošina automātisko logrīku atjauninšanu.
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninašanas pieprasījumus un izsauc onUpdate funkciju.
void	onUpdate(Context context, AppWidgetManager, appWidgetManager, int[] appWidgetIds)	Nolasa visu logrīku ID un katram no tiem izsauc funkciju widgetUpdate, kas veic logrīku atjauninašanu.
void	widgetUpdate(Context context,int widgetID)	Funkcija, kas veic logrīku atjauninašanu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
AppWidgetManager	appWidgetManager	Satur informāciju par visiem logrīkiem.
int	count	Satur konkrētā tipa logrīku

		skaitu.
String	currentDate	Datums.
String	caps	Mazie vai lielie burti.
int	fontSize	Burtu izmērs.
int[]	ids	Masīvs ar visiem konkrētā tipa logrīku ID numuriem.
RemoteViews	views	Norāde uz galveno logrīka skatu.
String	fontColor	Burtu krāsa.
String	UPDATE_WIDGETS	Logrīka atjauninšanas pieprasījuma teksts.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.7.4. Logrīka atjauninašnas pieprasījuma apstrādes moduļa projektējums

4.7.4.1. Klase DateWidgetAlarmManager

Klase, kas nodrošina periodisku logrīka datu atjauninašanu, pēc klasē noteiktā laika, nosūtot atjauninašanas pieprasījumu galvenajai logrīka klasei WidgetProvider.

Metodes:

Tips	**Metode**	**Apraksts**
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninājumu pieprasījumu.
void	setAlarm(Context context)	Funkcija, kas uzstāda logrīku automātisko atjaunošanos pēc noteikta laika intervāla.
void	clearAlarm(Context context)	Funkcija, kas noņem

	logrīku automātisko atjaunināšanos.

Mainīgie:

Tips	Mainīgais	Apraksts
String	ALARM_ACTION	Logrīka atjaunināšanas pieprasījuma teksts.
long	UPDATE_FREQUENCY	Logrīka atjaunināšanas biežums.
String	action	Satur informāciju par saņemto pieprasījumu.
Intent	updateIntent	Satur norādi uz DateWidgetAlarmManager klasi.
AlarmManager	alarmManager	Sistēmas servisa objekts.
PendingIntent	pIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.8.JiraReaderWidget moduļu projektējums

4.8.1. Logrīka konfigurācijas moduļa projektējums

4.8.1.1. Klase Configure

Klase izveido konfigurācijas skatu, kurš dod iespēju lietotājam norādīt kādu logrīka burtu izmēru un krāsu viņš vēlas, pēc šo datu ievades klase saglabā šos uzstādījumus izmantojot klases JiraReaderWidgetSharedPreferences metodes un izsauc WidgetProvider klasi, kas veic tālāku šo datu apstrādi un nepieciešamo papildklašu izsaukšanu.

Metodes:

Tips	Metode	Apraksts
void	OnCreate(savedInstanceState)	Ielādē konfigurācijas aktivitātes skatu un

		inicializē mainīgo widgetID.
void	onClick(View v)	Veic lietotāja konfigurāciju datu priekšapstrādi.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
Bundle	extras	Logrīka izveides brīdī izveidoto datu kopa.
Configure	context	Konfigurācijas aktivitātes konteksts.
int	widgetID	Satur tikko izveidotā logrīka ID.
Spinner	fontColorButton	Norāde uz burtu krāsas izvēles pogas skatu.
EditText	fontSizeButton	Norāde uz burtu izmēra ievades lauka skatu.
Button	acceptButton	Norāde uz ‘accept’ pogas skatu.
int	fontSize	Burtu izmērs.
String	fontColor	Burtu krāsa.
RemoteViews	views	Norāde uz galveno logrīka skatu.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
Intent	resultValue	Mainīgais, kurš glabā norādi uz WidgetProvider klasi.

4.8.2. Logrīka koplietojamo uzstādījumu moduļa projektējums

4.8.2.1. Klase JiraReaderWidgetSharedPreferences

Šī klase satur metodes, kas veic lietotāja ievadīto konfigurācijas datu saglabāšanu, nolasīšanu un dzēšanu no logrīka koplietojamo uzstādījumu XML formāta faila.

Metodes:

Tips	Metode	Apraksts
void	savePreferences(int FontSize, String FontColor, String Caps, int widgetID, Context context)	Fukcija, kas saglabā lietotāja norādītos iestatījumus.
int	getFontSize(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu izmēru.
String	getFontColor(int widgetID, Context context)	Funkcija, kas atgriež logrīka burtu krāsu.
void	deletePreferences(int widgetID, Context context)	Funkcija, kas izdzēš logrīka koplietojamo uzstādījumu ierakstus no to XML faila.
void	savePostNumber(int number, Context context)	Fukcija, kas saglabā dienas citāta numuru.
void	saveDate(String date, Context context)	Fukcija, kas saglabā datumu.
int	getPostNumber(Context context)	Fukcija, kas atgriež dienas citāta numuru.
String	getDate(Context context)	Fukcija, kas atgriež datumu.

Mainīgie:

Tips	Mainīgais	Apraksts

Editor	editor	Mainīgais, kas satur metodes, kas ļauj veikt izmaiņas koplietojamo uzstādījumu XML failā.
int	fontSize	Logrīka burtu izmērs.
String	fontColor	Logrīka burtu krāsa.
String	date	Datums.
int	number	Dienas citāta kārtas numurs.

4.8.3. Logrīka servisa moduļa projektējums

4.8.3.1. Klase WidgetProvider

Galvenā logrīka klase, kas atbildīga par logrīka datu pareizu attēlošanu tā galvenajā skatā un to periodisku atjaunināšanu. Šī klase nepieciešamības gadījumā var izsaukt visas pārējās logrīka klases.

Metodes:

Tips	Metode	Apraksts
void	onDeleted(Context context, int[] appWidgetIds)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie konkrēta logrīka izdzēšanas.
void	onDisabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie visu logrīka izdzēšanas. Izsauc, JiraReaderWidgetAlarmManager klases metodi clearAlarms, kas pārtrauc automātisko logrīku atjauninšanu.
void	onEnabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie pirmā logrīka novietošanas uz ierīces ekrāna vai ierīces

		pārstartēšanas. Izsauc, JiraReaderWidgetAlarmManager klases metodi setAlarms, kas nodrošina automātisko logrīku atjauninšanu.
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjaunināšanas pieprasījumus un izsauc onUpdate funkciju.
void	onUpdate(Context context, AppWidgetManager, appWidgetManager, int[] appWidgetIds)	Nolasa visu logrīku ID un katram no tiem izsauc funkciju widgetUpdate, kas veic logrīku atjaunināšanu.
void	widgetUpdate(Context context,int widgetID)	Funkcija, kas veic logrīku atjaunināšanu.
boolean	netConnect(Context ctx)	Funkcija, kas pārbauda vai Android ierīcei ir pieejams interneta savienojums.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
AppWidgetManager	appWidgetManager	Satur informāciju par visiem logrīkiem.
int	count	Satur konkrētā tipa logrīku skaitu.
int	fontSize	Burtu izmērs.
int[]	ids	Masīvs ar visiem konkrētā tipa logrīku ID numuriem.
RemoteViews	views	Norāde uz galveno logrīka skatu.
String	fontColor	Burtu krāsa.
String	UPDATE_WIDGETS	Logrīka atjauninšanas pieprasījuma teksts.

String	url	Tīmekļa adrese, no kuras tiek iegūta informācija par laikapstākļiem.
NetworkInfo	info	Informācija par to vai ierīcei pieejams Interneta pieslēgums.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
ConnectivityManager	cm	Mainīgais, kurš satur informāciju par ierīces tīkla pieslēgumu.

4.8.4. Logrīka atjaunināšnas pieprasījuma apstrādes moduļa projektējums

4.8.4.1. Klase JiraReaderWidgetAlarmManager

Klase, kas nodrošina periodisku logrīka datu atjaunināšanu, pēc klasē noteiktā laika, nosūtot atjaunināšanas pieprasījumu galvenajai logrīka klasei WidgetProvider.

Metodes:

Tips	**Metode**	**Apraksts**
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninājumu pieprasījumu.
void	setAlarm(Context context)	Funkcija, kas uzstāda logrīku automātisko atjaunošanos pēc noteikta laika intervāla.
void	clearAlarm(Context context)	Funkcija, kas noņem logrīku automātisko atjaunināšanos.

Mainīgie:

Tips	Mainīgais	Apraksts
String	ALARM_ACTION	Logrīka atjaunināšanas pieprasījuma teksts.
long	UPDATE_FREQUENCY	Logrīka atjaunināšanas biežums.
String	action	Satur informāciju par saņemto pieprasījumu.
Intent	updateIntent	Satur norādi uz DateWidgetAlarmManager klasi.
AlarmManager	alarmManager	Sistēmas servisa objekts.
PendingIntent	pIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.8.5. Logrīka Jira serviss datu apstrādes moduļa projektējums

4.8.5.1. Klase CustomAsyncTask

Klase, kas nodrošina logrīka vārdadienu datu apstrādi un to iegūšanas pieprasījuma nosūtīšanu klasei RSSHandler.

Metodes:

Tips	Metode	Apraksts
konstruktors	CustomAsyncTask(RemoteViews views, int appWidgetId, AppWidgetManager appWidgetManager, Context context)	Inicializē visus nepieciešamos mainīgos.
String	doInBackground(String... params)	Fona process, kurā tiek veikta servera faila apstrāde.

void	onPreExecute()	Funkcija, kura pirms faila apstrādes no skaidro vai ir jāmaina dienas citāts.
void	onPostExecute(String message)	Funkcija, kura pēc faila apstrādes atjaunina logrīka dienas citātu.

Mainīgie:

Tips	**Metode**	**Apraksts**
Context	context	CustomAsyncTask klases konteksts.
SimpleDateFormat	dateFormat	Nosacījumi, kādos formatēt datumu.
String	lastDate	Pēdējā logrīka atjaunināšanas brīža datums.
RemoteViews	views	Norāde uz galveno logrīka skatu.
int	widgetID	Logrīka identifikācijas numurs.
AppWidgetManager	widgetManager	Satur informāciju par visiem logrīkiem.
RSSHandler	rssHandler	RSSHandler klases objekts.
String	currentDate	Pašreizējais Android iekārtas datums.
URL	rssUrl	Apstrādājamā jiras servisa faila tīmekļa adrese.
SAXParserFactory	factory	Faila apstrādes objekts.
SAXParser	saxParser	Faila apstrādes objekta metode.

XMLReader	xmlReader	Faila apstrādes objekta metode.
InputSource	inputSource	Ievades plūsma.

4.8.5.2. Klase RSSHandler

Klase, kas veic datu nolasīšanu no Jira servera, šīs klases datu sagatavošanu un iegūto datu pēcapstrādi veic klase CustomAsyncTask.

Metodes:

Tips	**Metode**	**Apraksts**
void	startElement(String uri, String localName, String qName, Attributes a)	Funkcija, kas meklē sākuma tagu.
void	endElement(String uri, String localName, String qName)	Funkcija, kas meklē beigu tagu.
void	endDocument()	Funkcija, kas veic datu apstrādi, kad sasniegtas dokumentu beigas.
void	characters(char[] ch, int start, int length)	Funkcija, kas veic apstrādājamā faila simbolu apstrādi.

Mainīgie:

Tips	**Metode**	**Apraksts**
int	Counter	Mainīgais, kurš satur konkrētā apstrādājamā citāta kārtas numuru.
int	needed	Konkrētās dienas citāta kārtas numurs.
String[]	rssResult	Apstrādājamā teksta apstrādes brīža glabātuve.
StringBuilder	chars	Papildmainīgais vieglākai

teksta apstrādei.

4.9.TextEditorWidget moduļu projektējums

4.9.1. Logrīka konfigurācijas moduļa projektējums

4.9.1.1. Klase Configure

Klase izveido konfigurācijas skatu, kurš dod iespēju lietotājam norādīt kādu logrīka burtu izmēru un krāsu viņš vēlas, pēc šo datu ievades klase saglabā šos uzstādījumus izmantojot klases TextEditorWidgetSharedPreferences metodes un izsauc WidgetProvider klasi, kas veic tālāku šo datu apstrādi un nepieciešamo papildklašu izsaukšanu.

Metodes:

Tips	Metode	Apraksts
void	OnCreate(savedInstanceState)	Ielādē konfigurācijas aktivitātes skatu un inicializē mainīgo widgetID.
void	onClick(View v)	Veic lietotāja konfigurāciju datu priekšapstrādi vai veic nepieciešamās izmaiņas pēc kādas formatēšanas pogas nospiešanas.(Tādas šajā klasē ir 6, 5 no tām formatēšanas stiliem, bet viena „Accept” pogai).
void	afterTextChanged(Editable s)	Veic burtu apstrādi pēc konkrētā simbola ievadīšanas.
void	onItemSelected(AdapterView<?> parentView, View selectedItemView,	Funkcija, kura veic nepieciešamās izmaiņas

	int position, long id)	pēc pogas atlasītā elementa nomaiņas.(Tādas šajā klasē ir divas burtu krāsai un izmēram)
SpannableStringBuilder	trimSpannable(SpannableStringBuilder spannable)	Funkcija, kura pirms konfigurācijas datu atkārtotas ielādes noņem tukšuma zīmes no teksta beigām.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
Configure	context	Konfigurācijas aktivitātes konteksts.
int	widgetID	Satur tikko izveidotā logrīka ID.
int	boldStart	Treknraksta stila sākuma atzīmes atrašanās vieta.
int	italicStart	Slīpraksta stila sākuma atzīmes atrašanās vieta.
int	underlineStart	Pasvītrojuma stila sākuma atzīmes atrašanās vieta.
int	strikeStart	Pārsvītrojuma stila sākuma atzīmes atrašanās vieta.
int	colorStart	Burtu krāsas sākuma atzīmes atrašanās vieta.
int	sizeStart	Burtu izmēra sākuma atzīmes atrašanās vieta.
int	cursorLoc	Kursora atrašanās vieta.
int	prevGarums	Teksta garums pirms izmaiņas.
Bundle	extras	Logrīka izveides brīdī

		izveidoto datu kopa.
String	htmls	Teksts html formātā.
EditText	Text	Norāde uz teksta ievades skatu.
SpannableStringBuilder	spanned	
int	garums	Teksta garums pēc izmaiņas.
Spinner	spnColors	Norāde uz krāsu izvēles pogu.
Button	bColor	Norāde uz krāsas piešķiršanas pogu.
ToggleButton	strikeButton	Norāde uz pārsvītrojuma pogu.
ToggleButton	underlineButton	Norāde uz pasvītrojuma pogu.
ToggleButton	italicButton	Norāde uz pasvītrojuma pogu.
ToggleButton	boldButton	Norāde uz treknraksta pogu.
Spinner	spnSize	Norāde uz burtu izmēra izvēles pogu.
int[]	retrieve	Masīvs, kurā glabājas visu pieejamo krāsu identifikācijas numuri.
String[]	izmeri	Masīvs, kurā glabājas pieejamo teksta izmēru
int	colorPos	Izvēlētās krāsas kārtas numurs.
String	fontSizePirms	Burtu izmērs String datu tipā, tikko nolasīts no koplietojamajiem uzstādījumiem.
int	fontSizePec	Burtu izmērs Integer datu tipā.

int	selectionStart	Iezīmētā teksta sākuma pozīcija.
int	selectionEnd	Iezīmētā teksta beigu pozīcija.
Spannable	str	Teksts formatēšanas formātā.
ForegroundColorSpan[]	ss	Masīvs, kurā īslaicīgi uzglabā burtu krāsas stilus.
int	temp	Mainīgais īzlaicīgai citu mainīgo uzglabāšanai.
int	start	Stila sākuma pozīcija.
int	end	Stila beigu pozīcija.
CharacterStyle[]	appliedStyles	Masīvs kurā īslaicīgi uzglabā visus stilus.
StrikethroughSpan	strikeSpan	Pārsvītrojuma stils.
StrikethroughSpan[]	ss	Masīvs, kurā īslaicīgi uzglabā pārsvītrojuma stilus.
StyleSpan	BoldSpan	Treknraksta stils.
StyleSpan[]	ss	Masīvs, kurā īslaicīgi uzglabā treknraksta un slīpraksta stilus.
StyleSpan	ItalicSpan	Slīpraksta stils.
boolean	exists	Mainīgais, kurš glabā atzīmi par to vai iezīmētajam tekstam jau ir piešķirts kāds no stiliem.
UnderlineSpan	UnderlineSpan	Pasvītrojuma stils.
StrikethroughSpan	StrikeSpan	Pārsvītrojuma stils.
ForegroundColorSpan	ColorSpan	Burtu krāsas stils.
AbsoluteSizeSpan	SizeSpan	Burtu izmēra stils.
UnderlineSpan[]	ss	
int	trimStart	Apgriežamā teksta sākuma pozīcija.

int	trimEnd	Apgriežamā teksta beigu pozīcija.
String	trimText	Apgriežamais teksts.
Spannable	textBefore	Teksts pirms tā pārveidošanas html formātā.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.
Intent	resultValue	Mainīgais, kurš glabā norādi uz WidgetProvider klasi.

4.9.1.2. Klase SpinnerAdapter

Klase, kas nodrošina krāsu pogas funkcionalitāti.

Metodes:

Tips	**Metode**	**Apraksts**
int	getCount()	Atgriež, pogas izvēles opciju skaitu.
konstruktors	SpinnerAdapter(Context context)	Inicializē masīvu, un ieraksta tajā visus elementus.
Object	getItem(int arg0)	Atgriež konkrētu elementu no izvēlnes masīva.
long	getItemId(int arg0)	Atgriež konkrēta izvēlnes masīva identifikācijas numuru.
View	getView(int pos, View view, ViewGroup parent)	Atgriež pogas skatu.

Mainīgie:

Tips	Metode	Apraksts
Context	context	Konteksta tipa mainīgais.
ArrayList<Integer>	colors	Krāsu saraksts.
int[]	retrieve	Masīvs, kurā uzglabā krāsu kārtas numurus.
LayoutInflater	inflater	Skatu apstrādes objekts.
TextView	txv	Norāde uz pogas skatu.

4.9.1.3. Klase MyHtmlTagHandler

Klase, kas nodrošina pareizu pārsvītrojuma(<strike>) stila realizāciju konvertējot logrīka tekstu no html formāta.

Metodes:

Tips	Metode	Apraksts
void	handleTag(boolean opening, String tag, Editable output, XMLReader xmlReader)	Funkcija, kura atrod html tekstā konkrēto tagu.
void	processStrike(boolean opening, Editable output)	Funkcija, kas apstrādā tekstu, kurš ir ietverts <strike> tagos.
Object	getLast(Editable text, Class kind)	Funkcija, kura atgriež formatējuma atzīmju beigu punktu.

Mainīgie:

Tips	Mainīgais	Apraksts
int	len	Apstrādājamā teksta garums.
int	where	Apstrādājamā teksta sākums.
Object	obj	Apstrādājamā teksta

	beigas.

4.9.2. Logrīka koplietojamo uzstādījumu moduļa projektējums

4.9.2.1. Klase TextWidgetSharedPreferences

Šī klase satur metodes, kas veic lietotāja ievadīto konfigurācijas datu saglabāšanu, nolasīšanu un dzēšanu no logrīka koplietojamo uzstādījumu XML formāta faila.

Metodes:

Tips	Metode	Apraksts
void	savePreferences(String Text, int widgetID, Context context)	Fukcija, kas saglabā lietotāja ievadīto tekstu.
String	getFontSizes(int widgetID, Context context)	Funkcija, kas atgriež burtu izmēru
String	getText(int widgetID, Context context)	Funkcija, kas atgriež lietotāja ievadīto tekstu.
int	getLength(int widgetID, Context context)	Funkcija, kas atgriež lietotāja ievadītā teksta garumu.
void	saveLength(int length, int widgetID, Context context)	Funkcija, kas atgriež lietotāja ievadītā teksta garumu.
void	saveFontSizes(String Text, int widgetID, Context context)	Funkcija, kas saglabā burtu izmēru.
void	deletePreferences(int widgetID, Context context)	Funkcija, kas izdzēš logrīka koplietojamo uzstādījumu ierakstus no to XML faila.

Mainīgie:

Tips	Mainīgais	Apraksts
Editor	editor	Mainīgais, kas satur metodes, kas ļauj veikt izmaiņas koplietojamo uzstādījumu XML failā.
String	text	Lietotāja ievadītais teksts.
String	fontSizes	Teksta burtu izmērs.
int	length	Teksta garums.

4.9.3. Logrīka servisa moduļa projektējums

4.9.3.1. Klase WidgetProvider

Galvenā logrīka klase, kas atbildīga par logrīka datu pareizu attēlošanu tā galvenajā skatā un to periodisku atjaunināšanu. Šī klase nepieciešamības gadījumā var izsaukt visas pārējās logrīka klases.

Metodes:

Tips	Metode	Apraksts
void	onDeleted(Context context, int[] appWidgetIds)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie konkrēta logrīka izdzēšanas.
void	onDisabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie visu logrīka izdzēšanas. Izsauc, TextEditorWidgetAlarmManager klases metodi clearAlarms, kas pārtrauc automātisko logrīku atjauninšanu.

void	onEnabled(Context context)	Android automātiski ģenerēta funkcija, kura tiek izsaukta pie pirmā logrīka novietošanas uz ierīces ekrāna vai ierīces pārstartēšanas. Izsauc, TextEditorWidgetAlarmManager klases metodi setAlarms, kas nodrošina automātisko logrīku atjauninšanu.
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjaunināšanas pieprasījumus un izsauc onUpdate funkciju.
void	onUpdate(Context context, AppWidgetManager, appWidgetManager, int[] appWidgetIds)	Nolasa visu logrīku ID un katram no tiem izsauc funkciju widgetUpdate, kas veic logrīku atjaunināšanu.
void	widgetUpdate(Context context,int widgetID)	Funkcija, kas veic logrīku atjaunināšanu.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
AppWidgetManager	appWidgetManager	Satur informāciju par visiem logrīkiem.
int	count	Satur konkrētā tipa logrīku skaitu.
int[]	ids	Masīvs ar visiem konkrētā tipa logrīku ID numuriem.
RemoteViews	views	Norāde uz galveno logrīka skatu.
String	UPDATE_WIDGETS	Logrīka atjauninšanas pieprasījuma teksts.
String	sizeBeforeParse	Burtu izmērs, tikko nolasīts no koplietojamajiem

		uzstādījumiem, String datu tipā.
int	size	Burtu izmērs.
Intent	configureIntent	Mainīgais, kurš glabā norādi uz Configure klasi.
PendingIntent	pendConfigureIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.9.4. Logrīka atjaunināšnas pieprasījuma apstrādes moduļa projektējums

4.9.4.1. Klase TextWidgetAlarmManager

Klase, kas nodrošina periodisku logrīka datu atjaunināšanu, pēc klasē noteiktā laika, nosūtot atjaunināšanas pieprasījumu galvenajai logrīka klasei WidgetProvider.

Metodes:

Tips	**Metode**	**Apraksts**
void	onReceive(Context context, Intent intent)	Funkcija, kas saņem logrīka atjauninājumu pieprasījumu.
void	setAlarm(Context context)	Funkcija, kas uzstāda logrīku automātisko atjaunošanos pēc noteikta laika intervāla.
void	clearAlarm(Context context)	Funkcija, kas noņem logrīku automātisko atjaunināšanos.

Mainīgie:

Tips	Mainīgais	Apraksts
String	ALARM_ACTION	Logrīka atjaunināšanas pieprasījuma teksts.
long	UPDATE_FREQUENCY	Logrīka atjaunināšanas biežums.
String	action	Satur informāciju par saņemto pieprasījumu.
Intent	updateIntent	Satur norādi uz DateWidgetAlarmManager klasi.
AlarmManager	alarmManager	Sistēmas servisa objekts.
PendingIntent	pIntent	Mainīgais, kurš nosaka izpildāmo darbību, pie tā izsaukšanas.

4.10. InfoDesk moduļu projektējums

4.10..5.1. Klase ZwidgetToDrive

Klase, kas saskaita cik dažāda tipa logrīku pašlaik atrodas InfoDesk lietotnē un visus nepieciešamos sagatavošanas darbu lai logrīka koplietojamo uzstādījumu XML failus varētu veiksmīgi pārvietot uz GoogleDrive disku.

Metodes:

Tips	Metode	Apraksts
void	WidgetToDrive()	Funkcija, kura saskaita cik logrīku ir Infodesk lietotnē.
int	WidgetPreferences()	Funkcija, kas atgriež skaitli, ja skaitlis ir lielāks par 0, tad kādam logrīka vēl ir nepieciešams

		pārvietot koplietojamo uzstādījumu XML failu.
void	WidgetIntents()	Funkcija, kas veic nepieciešamo mainīgo inicializāciju, lai Infodesk lietotne varētu pārvietot logrīku konfigurāciju failus uz GoogleDrive disku.
konstruktors	ZWidgetToDrive(Context mContext)	Inicializē konteksta mainīgo.

Mainīgie:

Tips	**Mainīgais**	**Apraksts**
Context	context	Konteksta tipa mainīgais.
String	JIRA	JiraReaderWidget paketes nosaukums.
String	TEXTEDITOR	TextEditorWidget paketes nosaukums.
String	WEATHER	WeatherWidget paketes nosaukums.
String	DATE	DateWidget paketes nosaukums.
String	VARDADIENU	VardadienuWidget paketes nosaukums.
int	jiraWidget	Skaitītājs, kurš pārbauda vai Infodesk lietotne ir JiraReaderWidget logrīks.
int	textWidget	Skaitītājs, kurš pārbauda vai Infodesk

		lietotne ir TextEditorWidget logrīks.
int	weatherWidget	Skaitītājs, kurš pārbauda vai Infodesk lietotne ir WeatherWidget logrīks.
int	dateWidget	Skaitītājs, kurš pārbauda vai Infodesk lietotne ir DateWidget logrīks.
int	nameWidget	Skaitītājs, kurš pārbauda vai Infodesk lietotne ir VardadienuWidget logrīks.
int	checkWidgets	Palīgmainīgais, kurš nosaka vai visas logrīku konfigurācijas jau ir pārvietotas.
int	widgetCounter	Palīgmainīgais, kurš nosaka vai visas logrīku konfigurācijas jau ir pārvietotas.
SharedPreferences	appSharedPrefs	Mainīgais, kurš dod iespēju piekļūt Infodesk koplietojamajiem uzstādījumiem.
int	widgetNum	Logrīku skaits Infodesk lietotnē.
Context	widgetContext	Logrīku konteksts.

5. PROJEKTA ORGANIZĀCIJA

Projektā tika izvēlēta spējās programmēšanas (Agile) pieeja (10. avots), jo projekta sākumā nebij iespējams noskaidrot visas nepieciešamās prasības tā izstrādei. Programmatūras plānošanas un izstrādes laikā tā tika regulāri atrādīta klientiem, tādējādi noskaidrojot papildus prasības, no klientu lietotājstāstiem.

Neraugoties uz to, ka spējā izstrāde neparedz, ka lietotājstāstu testēšano veic izstrādātājs, tā kā pasūtītāja kā tāda nebija, tad testēšanu veica pats izstrādātājs, ik pēc laika pārbaudot lietotnes ar vienībtestiem.

Programmatūra tika izstrādāta pēc Android labās programmēšanas principiem, klases un skatu izkārtojumu nosaukumi tika veidoti pēc iepriekš nodefinētiem principiem, lai citam izstrādātājm pirmo reizi skatoties uz projektu būtu vieglāk to saprast.

Programmatūras izstrādē tika izmantota Java programmēšanas valoda.

5.1. Konfigurācijas pārvaldība

Lai kontrolētu konfigurāciju tika lietota Mercurial versiju kontroles sistēma ar Source Tree versiju kontroles sistēmas pārvaldības grafiskās saskarnes rīku.

Stabilai konfigurācijas kontrolei izmantoja Branch metodi, kuras stumbrā atradās strādājoša lietotne, savukārt stumbra zari - papildus lietotnes iespējas - tika vēlāk pievienoti pamatam - stumbram.

5.2. Kvalitātes nodrošināšana

Lai nodrošinātu kvalitāti:

- Izstrāde tika veikta pēc Objektorientētās programmēšanas principiem.
- Lietotņu vienībtestēšana tika veikta ne retāk, kā divreiz dienā.
- Izstrādes vidē koda noformatēšanā tika izmantots Android standarts.
- Izmaiņas tika nodotas konfigurācijas pārvaldības sistēmai katras darba dienas beigās.
- Lietotnes tika notestētas uz vairākām ierīcēm un Android versijām.

6. DARBIETILPĪBAS NOVĒRTĒJUMS

Darbietilpības novērtējums tika izstrādāts projekta izpildīšanas sākumā, pamatojoties uz katra lietotājstāsta novērtēšanu, izmantojot Fibonači punktu skalu (5. avots). Ar 2, 3, 5 punktiem tika novērtēti tie lietotājstāsti, kurus izstrādātājs uzskatīja par vieglākiem, 8, 13, 21 punktiem tika novērvērtēti laikietilpīgākie. Kopsummā visi lietotājstāstu sarežģītība tika novērtēta 107 punktos. Tabulā 6.1. redzams sarežģītības punktu sadalījums pa iterācijām.

6.1. tabula

Sarežģītības punktu sadalījums pa iterācijām Iterācijas	**Sarežģītības punkti**
Pirmā	20
Otrā	21
Trešā	29
Ceturtā	29
Piektā	8

Izstrādes ātrums tika pieņemts projekta sākumā, balstoties uz izstrādātāja pieredzi - 7 punkti nedēļā, līdz ar to visam projektam ir nepieciešamas apmēram 15 nedēļas. Daži lietotājstāsti neaizņēma tik daudz laika, cik tam bija atvēlēts, tāpēc izstrādes laikā lietotnes tika papildinātas ar papildus iespējām, kuras lietotājstātos nebija pieminētas.
Vidējais izstrādes ātrums bija 7,13 punkti nedēļā - to izstrādātājs aprēķināja projekta beigās.

7. TESTĒŠANAS DOKUMENTĀCIJA

Testēšana tika nodrošināta izmantojot vienībtestus. Pēc katras reizes, kad programmētājs lietotni papildināja ar jaunām funkcijām, tika veikta pārbaude vai tas nav izmainījis iepriekšējo funkciju izpildi.

7.1. Pirmās iterācijas vienībtestēšana

Lietotājstāsta numurs	Testa ID	Uzdevums	Gaidāmais rezultāts	Izpilde
1.	7.1.1.	1.1.Pievienot visiem logrīkiem konfigurācijas aktivitāti, kura ļaus lietotājam norādīt vēlamos iestatījumus.	Novietojot logrīkus uz sākuma ekrāna, vai InfoDesk lietotnē, kā pirmais skats parādās konfigurācijas aktivitātes skats.	+
	7.1.2.	1.2.Izveidot XML dizaina failu konfigurācijas aktivitātes grafiskajai saskarnei.	Konfigurācijas aktivitātes skats atveras kā dialoglogs, kurā lietotājam parādās katram logrīkam nepieciešamās uzstatījumu pogas.	+
	7.1.3.	1.3.Izveidos aktivitāti, kas saglabās lietotāja ievadītos iestatījumus XML formāta	Pēc logrīka izveides visi lietotāja ievadītie iestatījumi tiek saglabāti XML formāta failā.	+

		failos.		
	7.1.4.	1.4.Izveidot metodi, kas iestatīs lietotāja ievadītos datus konkrētajam logrīkam.	Izveidotā logrīka izskats sakrīt ar lietotāja ievadītajiem uzstadījumiem.	+
2.	7.1.5.	2.1.Izveidot izsaukumu, kurš lietotājam pieskaroties logrīka grafiskajai saskarnei atkārtoti izsauks logrīka konfigurācijas aktivitāti.	Lietotāja uzklikšķinot uz logrīka skata jebkurā vietā atkārtoti atveras konfigurācijas aktivitāte.	+(ja lietotājs uzklikšķina uz logrīka teksta lauka)
	7.1.6.	2.2.Pievienot konfigurācijas aktivitātei trigeri, kurš saņemot izsaukumu atvērs konfigurācijas aktivitāti.	Kad lietotājs ir atkārtoti atvēris konfigurācijas aktivitātes skatu tas tiek korekti attēlots, jeb izskatās tieši tāpat, kā iepriekšējajā tā atvēršanas brīdī.	+
4.	7.1.7.	4.1.Veidot logrīkus nevis kā peldošos logus „InfoDesk” lietotnē, bet gan kā atsevišķas	Logrīku var novietot, gan uz Android ierīces sākuma ekrāna, gan InfoDesk lietotnē un tas ir	+

		programmas ar atbilstošām klasēm.	redzams ierīces logrīku izvēlnē.	

7.2. Otrās iterācijas vienībtestēšana

Lietotājstāsta numurs	Testa ID	Uzdevums	Gaidāmais rezultāts	Izpilde
3.	7.2.1	3.1.Pievienot visiem logrīkiem AlarmManager klasi, kas dos iespēju veikt periodiskus atjauninājumus.	Logrīka onEnabled metode izsauc automātisko logrīku atjaunināšanu ne retāk kā reizi 10 minūtēs.	+(atsevišķiem logrīkiem automātiskā atjaunināšana tiek veikta pārāk bieži jeb reizi 10 sekundēs)
	7.2.2	3.2.Pievienot AlarmManager klasei metodes, kas ļauj izsaukt un noņemt periodiskos atjauninājumus.	Pēc visu logrīku izdzēsšanas atjaunināšanas serviss beidz savu darbību un pēc ierīces pārstartēšanas, atjaunināšanas serviss sākas no jauna(ja kāds no logrīkiem ir aktīvs).	+
	7.2.3.	3.3.Iestatīt automātisko atjauninājumu servisa sākšanos pie logrīka izveides.	Pēc tam, kad logrīks ir izveidots, tas tiek periodiski atjaunots ne retāk kā reizi 10	+(atsevišķiem logrīkiem automātiskā atjaunināšana tiek veikta

			minūtēs.	pārāk bieži jeb reizi 10 sekundēs)
5.	7.2.4.	5.1.Realizēt visu logrīku iestatījumu klases funkcionalitāti tā lai katram logrīga identifikācijas numuram būtu saglabāti atbilstošie uzstādījumi.	Logrīka koplietojamo uzstādījumu XML failā pretī, katriem iestatījumiem ir redzams logrīka identifikācijas numurs, kuram logrīkam šie uzstādījumi atbilst.	+
	7.2.5.	5.2.Izveidot metodes, kas ļauj iegūt katra logrīka uzstādījumus un tos iestatīt konkrētajam logrīkam.	Katram logrīkam ir izveidota SharedPreferences klase, kuras metodes atgriež nepieciešamos uzstādījumus no koplietojamo uzstādījumu XML faila.	+
6.	7.2.6.	6.1.Pārveidot „JiraReaderWidget” logrīka klases, kas atbild par datu nolasi tā, lai katru dienu no servera tiktu nolasīts atbilstošais citāts.	Katru dienu pēc pulksten 24:00(pēc Android ierīces laika) logrīka dienas citāts tiek nomainīts.	+(taču iespējama aizkave līdz 10 sekundēm)
	7.2.7.	6.2.Pievienot papildus pogu „DateWidget” konfigurācijas skata	Atverot „DateWidget” logrīka konfigurācijas	+

		XML failā , kas ļaus iestatīt to vai logrīka teksts sastāv no mazajiem vai lielajiem burtiem un attiecīgi realizēt tās funkcionalitāti.	aktivitātes skatu parādās papildus poga, kura ļauj iespēju lietotājam izvēlēties vai attēlot logrīku ar lielajiem vai mazajiem burtiem.	

7.3. Trešās iterācijas vienībtestēšana

Lietotājstāsta numurs	Testa ID	Uzdevums	Gaidāmais rezultāts	Izpilde
9.	7.3.1.	9.1.Izveidot logrīku pamata funkcionalitāti atbilstoši pārējo logrīku shēmai.	No lietotāja skata punkta „TextEditorWidget" logrīks sastāv no visiem tiem pašiem skatiem, kuri ir pārējajiem logrīkiem(vienīgi konfigurācijas aktivitātes skats un galvenais logrīka skats ir atšķirīgi).	+
	7.3.2.	9.2.Pievienot visus nepieciešamos logrīkas skatu XML failus.	Logrīkam ir konfigurācijas skats un galvenais skats.	+
		9.3.Pievienot logrīka konfigurācijas aktivitātei teksta	Konfigurācijas aktivitātes skatā ir teksta lauks, kurā lietotājs var ievadīt	+

		lauku, kurā lietotājs ievadīs tekstu.	tekstu paša izvēlētā formatējumā.	
10.	7.3.3.	10.1.Izveidot izsaukumu, kurš lietotājam pieskaroties logrīka grafiskajai saskarnei atkārtoti izsauks logrīka konfigurācijas aktivitāti.	Lietotāja uzklikšķinot uz logrīka skata jebkurā vietā atkārtoti atveras konfigurācijas aktivitāte.	+
	7.3.4.	10.2.Pievienot konfigurācijas aktivitātei trigeri, kurš saņemot izsaukumu atvērs konfigurācijas aktivitāti.	Kad lietotājs ir atkārtoti atvēris konfigurācijas aktivitātes skatu tas tiek korekti attēlots, jeb izskatās tieši tāpat, kā iepriekšējajā tā atvēršanas brīdī.	+

7.4. Ceturtās iterācijas vienībtestēšana

Lietotājstāsta numurs	Testa ID	Uzdevums	Gaidāmais rezultāts	Izpilde
11.	7.4.1.	11.1.Izmainīt logrīka konfigurācijas aktivitātes XML failu, pievienojot, tam papildus funkcionalitātes pogas(teksta krāsas un burtu izmēra	Konfigurācijas aktivitātes skatam ir pievienotas pogas burtu krāsas un izmēra nomaiņai.	+

		nomaiņai).		
	7.4.2	11.2.Realizēt visu šo papildus pogu funkcionalitāti (teksta krāsas un burtu izmēra nomaiņai)logrīka konfigurācijas aktivitātē.	Uzspiežot uz teksta izmēra vai burtu krāsas pogas tekstam tiek nomainīta krāsa vai izmērs.	+
12.	7.4.3.	12.1.Izmainīt logrīka konfigurācijas aktivitātes XML failu, pievienojot, tam papildus funkcionalitātes pogas(treknrakstam, slīprakstam, pasvītrojumam un pārsvītrojumam).	Konfigurācijas aktivitātes skatam ir pievienotas pogas treknrakstam, slīprakstam, pasvītrojumam un pārsvītrojumam.	+
	7.4.4.	12.2.Realizēt visu šo papildus pogu(treknrakstam, slīprakstam, pasvītrojumam un pārsvītrojumam) funkcionalitāti logrīka konfigurācijas aktivitātē.	Uzspiežot uz kādas no šīm pogām tekstam tiek iestatīts atbilstošais stils.	+
13.	7.4.5.	13.1.Pievienot logrīkam „TextWatcher” funkciju, kas veiks katra atsevišķā	Lietotājam ir iespēja mainīt stilu katram atsevišķam teksta simbolam.	+(izmēra poga nenomaina katra simbola izmēru, bet visa teksta simbolu

		simbola apstrādi.		izmēru kopumā, sakarā ar logrīka darbības ātruma samazināšanos pie šīs opcijas pielikšanas)

7.5. Piektās iterācijas vienībtestēšana

Lietotājstāsta numurs	Testa ID	Uzdevums	Gaidāmais rezultāts	Izpilde
8.	7.5.1.	8.1.Izveidot klasi „InfoDesk" lietotnē, kuras uzdevums būs noteikt cik daudz logrīku pašlaik atrodas lietotnē.	InfoDesk lietotnē ir izveidota klase, kas precīzi nosaka to cik logrīku(atšķirīga tipa) ir Infodesk lietotnē.	+
	7.5.2.	8.2.Izveidot metode jau esošā „InfoDesk klasē", kuras uzdevums būs izsaukt katra logrīka konfigurāciju XML failu pārvietošanu uz GoogleDrive disku.	InfoDesk lietotnes galvenajā izvēlnē nospiežot export pogu, visu logrīku koplietojamo uzstādījumu XML faili tiek pārvietoti uz GoogleDrive disku.	+(tomēr pašlaik nav realizēta to veiksmīga pielietošana pēc to pārvietošanas atpakaļ)

SECINĀJUMI

Ņemot vērā to, ka autors iepriekš nebija nodarbojies ar Android lietotņu izstrādi pirms šī projekta uzsākšanas un tā sākuma posmā autoram bija jāiepazīstas ar Java programmēšanas valodu un Android raksturīgajām klasēm.Tomēr autors veiksmīgi spēja izpildīt lietotāja norādītās prasības iekļaujoties projekta izstrādei atvēlētajā laikā.
Izstrādes laikā autors iepazinās ne tikai ar Java programmēšanas valodas pamatprincipiem un Android operētājsistēmai raksturīgajām bibliotēkām, bet arī padziļināja savu izpratni par XML un Json formātu datu nolasi un apstrādi.
Nākotnē, balstoties uz šajā projektā iegūtajām zināšanām, autors plāno uzlabot šo projektu vairāk koncentrējoties uz logrīku datu koplietošanu starp dažādām Android ierīcēm.

IZMANTOTĀ LITERATŪRA

1. <uses-sdk>.[tiešaiste]. [Skatīts 15.03.2015]. Pieejams: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html
2. 3. А. Голощапов, Google Android Создание приложений для смартфонов и планшетных ПК, БХВ-Петербург, 2013
3. Android Activity as a dialog.[tiešsaiste].[Skatīts 12.03.2015]. Pieejams: http://stackoverflow.com/questions/1979369/android-activity-as-a-dialog
4. App Widgets. [tiešsaiste]. [Skatīts 03.03.2015]. Pieejams: https://developer.android.com/guide/topics/appwidgets/index.html
5. Darbietilpības prognozēšana [tiešsaiste]. [Skatīts 15.05.2015]. Pieejams: http://estudijas.lu.lv/pluginfile.php/258971/mod_resource/content/1/Darbietilpiibas%20prognozeeshana%20-%20Liva%20Steinberga%20-%2029%2010%202012.pdf
6. Draw.io. [tiešsaiste]. [Skatīts 03.05.2015]. Pieejams: https://www.draw.io
7. K. Upenieks, M. Aldiņš, R. Kļava, T. Lācis, T. Kirtovskis, O. Siliņš "Exigen InfoDesk tehniskā dokumentācija" 2015
8. Microsoft Visio 2013. [bezsaiste]. [Skatīts 03.04.2015]. Pieejams: https://e5.onthehub.com/WebStore/OfferingsOfMajorVersionList.aspx?pmv=8a7aa9f6-2034-e211-aed3-f04da23e67f6&cmi_mnuMain=bdba23cf-e05e-e011-971f-0030487d8897&ws=07fb2d32-836f-e011-971f-0030487d8897&vsro=8
9. SourceTree.[tiešsaiste]. [Skatīts 15.04.2015]. Pieejams: https://www.sourcetreeapp.com/
10. Spējā izstrāde [tiešsaiste]. [Skatīts 02.14.2014]. Pieejams: http://estudijas.lu.lv/mod/page/view.php?id=128545
11. Style EditText content 'on the fly'?.[tiešsaiste].[Skatīts 12.04.2015].Pieejams: http://stackoverflow.com/questions/3096332/style-edittext-content-on-the-fly
12. W. Frank Ableson, Robi Sen, Chris King, Android in action 2nd edition, 2011

PIELIKUMS

Visas DateWidget logrīka klases:

Klase Configure:

```
import android.app.Activity;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RemoteViews;
import android.widget.Spinner;
//class responsible for all widget configuration setting up
public class Configure extends Activity {
static Configure context;
private int widgetID;
//sets configure.xml file layout as this class user interface
//also reads widgetID variable which will indicate this widget through all
his life cycle
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.configure);
setResult(RESULT_CANCELED);
context=this;
Bundle extras = getIntent().getExtras();
if(extras!= null){
widgetID = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
                                AppWidgetManager.INVALID_APPWIDGET_ID);
                }
final Spinner fontColorButton= (Spinner)findViewById(R.id.color_input);
final Spinner capsButton = (Spinner)findViewById(R.id.caps_input);
final EditText fontSizeButton =
(EditText)findViewById(R.id.fontsize_input);
Button acceptButton= (Button) findViewById(R.id.button1);
acceptButton.setOnClickListener(new OnClickListener() {
```

```
//saves all data using DateWidgetAlarmManager class functions after user
have entered all configuration
//and calls WidgetProvider class witch will do rest of the work
@Override
public void onClick(View v) {
int fontSize= Integer.parseInt(fontSizeButton.getText().toString());
String fontColor = fontColorButton.getSelectedItem().toString();
String Caps = capsButton.getSelectedItem().toString();
DateWidgetSharedPreferences.savePreferences(fontSize,fontColor,Caps,
widgetID, context);
WidgetProvider.widgetUpdate(context,widgetID);
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.main);
Intent configureIntent = new Intent(context, Configure.class);
configureIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);

configureIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
configureIntent.setData(Uri.parse(configureIntent.toUri(Intent.URI_INTENT_S
CHEME)));
PendingIntent pendConfigureIntent = PendingIntent.getActivity(context,
widgetID, configureIntent, PendingIntent.FLAG_UPDATE_CURRENT);
views.setOnClickPendingIntent(R.id.text, pendConfigureIntent);
Intent resultValue = new Intent();
resultValue.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
setResult(RESULT_OK, resultValue);
finish();
}});
}}
```

Klase DateWidgetSharedPreferences:

```
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.preference.PreferenceManager;
//this class holds all functions that will save widget preferences in form:
Preference+widgetID+value
//Used MODE_WORLD_READABLE because InfoDesk software application will need
to use this widgets preferences
public class DateWidgetSharedPreferences {
```

```
public static void savePreferences(int FontSize, String FontColor, String
Caps, int widgetID, Context context) {
Editor editor=
context.getSharedPreferences("DateWidget",Context.MODE_WORLD_READABLE).edit
();
editor.putInt("FontSize" + widgetID, FontSize);
editor.putString("FontColor" + widgetID, FontColor);
editor.putString("Caps" + widgetID, Caps);
editor.commit();
                                                    }
public static int getFontSize(int widgetID, Context context)
{
SharedPreferences prefs = context.getSharedPreferences("DateWidget",
Context.MODE_WORLD_READABLE);
int fontSize = prefs.getInt("FontSize" + widgetID, 25);
return fontSize;
}
public static String getFontColor(int widgetID, Context context)
{
SharedPreferences prefs =
context.getSharedPreferences("DateWidget",Context.MODE_WORLD_READABLE);
String fontColor = prefs.getString("FontColor" + widgetID, "#FFFFFF");
return fontColor;
}
public static String getCaps(int widgetID, Context context)
{
SharedPreferences prefs = context.getSharedPreferences("DateWidget",
Context.MODE_WORLD_READABLE);
String caps = prefs.getString("Caps" + widgetID, "LoverCase");
return caps;
}
public static void deletePreferences(int widgetID, Context context) {
Editor editor =
context.getSharedPreferences("DateWidget",Context.MODE_WORLD_READABLE).edit
();
editor.remove("FontSize" + widgetID);
editor.remove("FontColor" + widgetID);
editor.remove("Caps" + widgetID);
editor.commit();
}
}
```

Klase DateWidgetAlarmManager:

```
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.SystemClock;
//this class sends intents to WidgetProvider class when update is needed
public class DateWidgetAlarmManager extends BroadcastReceiver{
public static final String ALARM_ACTION =
"com.exigenservices.home.hsdatetime.ALARM_ACTION";
//widget update period, which easly can be changed
private static final long UPDATE_FREQUENCY = (1000*10);
@Override
public void onReceive(Context context, Intent intent) {
String action = intent.getAction();
if(action.equals(DateWidgetAlarmManager.ALARM_ACTION))
{

Intent updateIntent = new Intent(WidgetProvider.UPDATE_WIDGETS);
                context.sendBroadcast(updateIntent);
}
}
//called by WidgetProvider onEnabled function when widget is created or
device is restarted
public static void setAlarm(Context context){
AlarmManager alarmManager=
(AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
Intent alarmIntent = new Intent(DateWidgetAlarmManager.ALARM_ACTION);
PendingIntent pIntent = PendingIntent.getBroadcast(context, 0, alarmIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                        SystemClock.elapsedRealtime()+
DateWidgetAlarmManager.UPDATE_FREQUENCY,
DateWidgetAlarmManager.UPDATE_FREQUENCY, pIntent);
      }
public static void clearAlarm(Context context){
```

```
AlarmManager alarmManager=
(AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
Intent alarmIntent = new Intent(DateWidgetAlarmManager.ALARM_ACTION);
PendingIntent pIntent = PendingIntent.getBroadcast(context, 0, alarmIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
            alarmManager.cancel(pIntent);
            }}
```

Klase WidgetProvider:

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.util.Log;
import android.widget.RemoteViews;
import android.widget.Toast;

//main widget class which is responsible for all widget updating after
their creation
public class WidgetProvider extends AppWidgetProvider
{
public static final String tag = "WidgetProvider";
public static final String UPDATE_WIDGETS
="com.exigenservices.home.hsdatetime.UPDATE_WIDGETS";
private static SimpleDateFormat dateFormat = new SimpleDateFormat("EEEE, d.
MMMM",  new Locale("lv"));
@Override
//periodically gets update intents from DateWidgetAlarmManager class
public void onReceive(Context context, Intent intent) {
super.onReceive(context, intent);
if(intent.getAction().equals(WidgetProvider.UPDATE_WIDGETS))
{AppWidgetManager appWidgetManager =AppWidgetManager.getInstance(context);
```

```
int[] ids = appWidgetManager.getAppWidgetIds(new ComponentName(context,
WidgetProvider.class));
onUpdate(context, appWidgetManager, ids);
}
}
@Override
public void onDeleted(Context context, int[] appWidgetIds)
{
super.onDeleted(context, appWidgetIds);
Toast.makeText(context, "Delete", Toast.LENGTH_LONG).show();
}
@Override
//clears alarm system service when all this type widgets are removed from
device
public void onDisabled(Context context)
{
super.onDisabled(context);
Toast.makeText(context, "Disable", Toast.LENGTH_LONG).show();
DateWidgetAlarmManager.clearAlarm(context);
}
@Override
//sets alarm system service when first widget is created or device have
just completed restart
public void onEnabled(Context context)
{
super.onEnabled(context);
Toast.makeText(context, "Enable", Toast.LENGTH_LONG).show();
            DateWidgetAlarmManager.setAlarm(context);}
@Override
//used method of getting all widget ids and than running update for each of
them with function widgetUpdate
public void onUpdate(
Context context,
AppWidgetManager appWidgetManager,
int[] appWidgetIds
)
{
super.onUpdate(context, appWidgetManager, appWidgetIds);
int count = appWidgetIds.length;
for(int i=0; i<count; i++)
{
WidgetProvider.widgetUpdate(context, appWidgetIds[i]);
```

```
}
}
//this function is responsible for updating each widget seperatly
public static void widgetUpdate(Context context,int widgetID)
{
AppWidgetManager manager = AppWidgetManager.getInstance(context);
int fontSize = DateWidgetSharedPreferences.getFontSize(widgetID, context);
String fontColor = DateWidgetSharedPreferences.getFontColor(widgetID,
context);
String caps = DateWidgetSharedPreferences.getCaps(widgetID, context);
String currentDate = dateFormat.format(Calendar.getInstance().getTime());
currentDate= currentDate.replace(". ", ".");
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.main);
if(caps.equals("LoverCase"))
{
views.setTextViewText(R.id.text, currentDate);
}
else if(caps.equals("UpperCase"))
{
views.setTextViewText(R.id.text, currentDate.toUpperCase(new
Locale("lv")));
}
views.setFloat(R.id.text,"setTextSize" ,fontSize);
views.setTextColor(R.id.text, Color.parseColor(fontColor));
Intent configureIntent = new Intent(context, Configure.class);
configureIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
configureIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
configureIntent.setData(Uri.parse(configureIntent.toUri(Intent.URI_INTENT_S
CHEME)));
PendingIntent pendConfigureIntent = PendingIntent.getActivity(context,
widgetID, configureIntent, PendingIntent.FLAG_UPDATE_CURRENT);
views.setOnClickPendingIntent(R.id.text, pendConfigureIntent);
manager.updateAppWidget(widgetID, views);
}
}
```

DateWidget galvenais(main.xml) skats:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
android:orientation="vertical" >
<TextView
     android:id="@+id/text"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:textSize="30sp"
     android:textColor="#FFFFFF"
     android:textStyle="bold"
     android:maxLines="2"/>
</LinearLayout>
```

Kvalifikācijas darbs „***Informatīvā paneļa "InfoDesk" konfigurējamu logrīku 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: ***Kaspars Upenieks*** _____ **___.05.2015**.

Rekomendēju darbu aizstāvēšanai

Darba vadītāj**s:Dr.inž. Artis Teilāns**_____ **___.05.2015.**

Recenzents: ***M.dat, M.ekon., MBA Andris Bozis***

Darbs iesniegts 01.06.2015.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): _____