LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Informatīvā paneļa "InfoDesk" tīkla funkcionaitātes izstrāde

KVALIFIKĀCIJAS DARBS

Autors: **Toms Kirtovskis**
Studenta apliecības Nr.: tk13015
Darba vadītājs: Dr.inž. Artis Teilāns

RĪGA 2015

ANOTĀCIJA

Toma Kirtovska kvalifikācijas darbā " Informatīvā paneļa "InfoDesk" tīkla funkcionaitātes izstrāde " ir izstrādāta Informatīvā paneļa "InfoDesk" tīkla funkcionaitāte.

Tīkla funkcionalitāte sniedz iespēju, izmantojot Google Drive piedāvātās bezmaksas funkcijas, augšupielādēt un lejupielādēt “InfoDesk” uzstādījumus un izmantot tos dažādās ierīcēs, kā arī sniedz sinhronizācijas iespējas, tādejādi nodrošinot “InfoDesk” informatīvā paneļa attēlotā satura regulāru atjauninašanu.

Lietotne ir izstrādātas Android vidē.

Atslēgas vārdi: Android, Agile, “Infodesk”, Google Drive.

ABSTRACT

This qualification paper “Development of network functionality for informative panel “InfoDesk”” is about development of network functionality for informative panel “InfoDesk”.

Network functionality provides import and export of “InfoDesk” settings to other devices using Google Drive applications. And includes synchronization options which provides automatic updates with selected frequency.

Application is designed for Android devices.

Key words: Android, Agile, “InfoDesk”, Google Drive.

Saturs

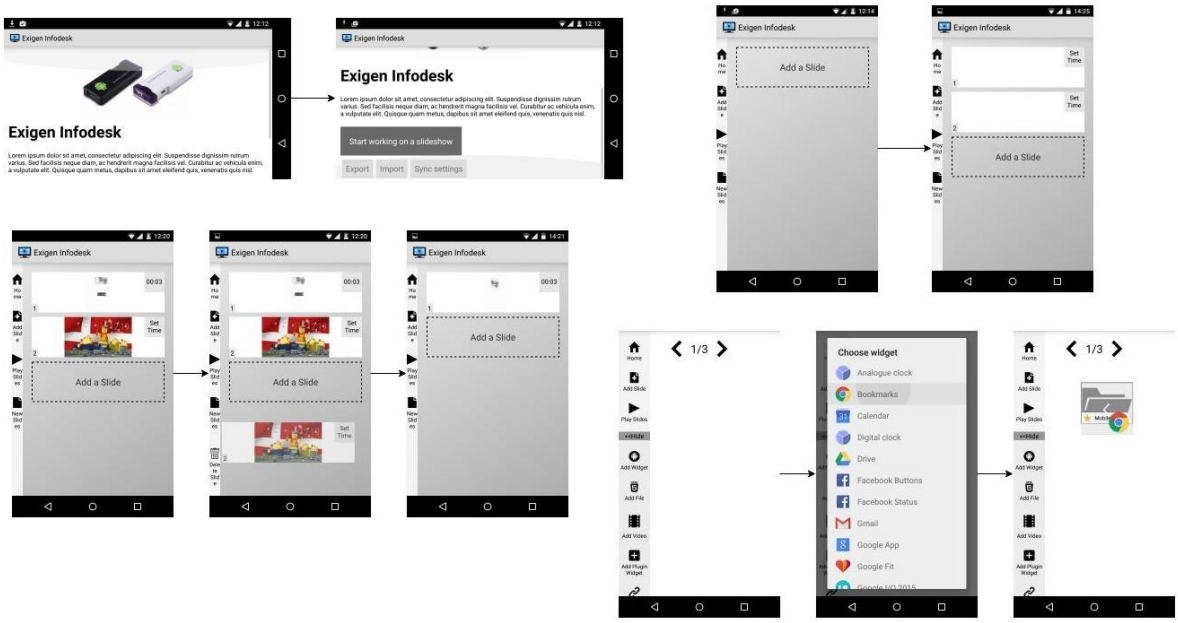
1. IEVADS

1.1. Nolūks

Šī dokumenta nolūks ir aprakstīt “InfoDesk” lietotnes tīkla funkcionalitātes prasības tas ir paredzēts moduļu projektētājam un pasūtītājam produkta funkcionalitātes prasību saskaņošanai.

1.2. Darbības sfēra

Šajā dokumentā ir definētas prasības Android lietotnes informatívā paneļa “InfoDesk” tīkla funkcionalitātei. “InfoDesk” paredzēts prezentāciju veidošanai. Attēlā 1.2.1. redzama “InfoDesk galvenā izvēlne”, slaidu pārlūkošanas skats (slaidu pievienošana un dzēšana), kā arī slaidu veidošanas skats.



Att.1.2.1.

Programmatūras produkta mērķis ir nodrošināt informatīvā paneļa “InfoDesk” attālinātu kontroli un tā uzstādījumu sinhronizāciju ar Google Drive. Produkta paredzētais pielietojums ir lietotnes pieejamība Google Play veikalā.

1.3. Definīcijas, akronīmi un saīsinājumi

Tabula 1.3.1. Definīcijas, akronīmi un saīsinājumi

Android	Mobilo ierīču operētājsistēma.
Android Studio	Android lietotņu izstrādes vide.
Java	Programmēšanas valoda.
Mercurial	Versiju kontroles sistēma.
SourceTree	Versiju kontroles sistēmas pārvaldības grafiskās saskarnes rīks.[9]
XML	Paplašināmā iezīmēšanas valoda.
Intent(intents)	Viens no Android operētājsistēmas interfeisiem, kas pārvalda signālus ar datu pielikumu, kuri tiek izmantoti, lai veidotu starp-programmatūru saziņu.
Google Drive	Google tiešsaistes failu uzglabāšanas serviss

2. VISPĀRĒJAIS APRAKSTS

2.1. Produkta nolūks

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]

2.2. Produkta perspektīva

“InfoDesk” paredzēts lietošanai dažādos un dažādu izmēru uzņēmumos, to ņemot vērā rodas nepieciešamība pēc iespējas

Produkta tīkla funkcionalitāte sniedz iespēju eksportēt vai importēt izveidoto prezentāciju no vienas ierīces uz citu, izmantojot Google Drive.

2.3. Produkta funkcijas

Programmatūra sniedz iespēju importēt jau sagatavotu prezentāciju no Google Drive, kā arī to eksportēt uz Google Drive, izveidojot vienu vai vairākas jaunas mapes vai vienkārši aizstājot jau esošu prezentāciju, kā arī uzstādīt sinhronizācijas intervālu, kuru ievērojot tiek veikta automātiska prezentācijas satura atjaunināšana, ja tas ir nepieciešams.

2.4. Lietotāja raksturiezīmes

Skolu audzēkņi, uzņēmumu un iestāžu darbinieki vai citi lietotāji, kuri ikdienā izvēlas lietot informatīvo paneli „InfoDesk”. Lietotājam nav jābūt ar īpašām priekšzināšanām.

2.5. Vispārējie ierobežojumi

Lietotnes izmantošanai ir nepieciešama Android ierīce ar API līmeni 14 vai augstāku, kā arī tīkla funkcionalitāte ir pieejama tikai, ja ir interneta pieslēgums un ir Google profils ar Google Drive funkcionalitāti.

3. LIETOTĀJA STĀSTI

Projekta sākumā tika nolemts programmatūru izstrādāt ar spējo (Agile [10]) programmēšanas metodoloģijas pieeju, tāpēc funkcionālās prasības tiek noteiktas balstoties uz „lietotāju stāstiem", kuri ņemot vērā to, ka nebija konkrētā pasūtītāja tika teorētiski radīti programmatūras izstrādes laikā.

Balstoties uz „lietotāju stāstiem", izstrādātājs varēja noteikt, ko vēlētos teorētiskais lietotājs un kāda ir katras prasības prioritāte attiecībā pret citām prasībām. „Lietotāja stāsti" ir izvietoti tabulas veidā, kurā ir attēloti visi lietotāja stāsti. Šie stāsti ir sakārtoti tādā secībā, kā tie tika radīti. To sarežģītība ir izteikta punktos, kurus izstrādātājs noteica, balstoties uz savām spējām un zināšanām par katras konkrētās prasības realizāciju.

3.1. Lietotājstāstu tabula

Tīkla funkcionalitātes lietotājstāsti ir attēloti tabulā 3.1.1.

3.1.1. Tabula Tīkla funkcionalitātes lietotājstāsti

Vispārējie tīkla funkcionalitātes stāsti		
Lietotāja stāsts	Prioritāte	Sarežģītības punkti
Iepazīšanās ar uzņēmuma iekšējo sistēmu, izstrādes vidi, valodu un projektu.	1	21
Nepieciešams eksportēt prezentācijas uzstādījumus no Android ierīces	1	21
Nepieciešams importēt prezentācijas uzstādījumus citā Android ierīcē	1	21
Fona attēlu eksports/imports	2	21
Automātiskas sinhronizācijas izveide	2	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 raksturo to, kā izpildītājs plāno šo prasību realizēt.

3.2.1. Iepazīšanās iterācija

Pirms darbu uzsākšanas ir jāiepazīstas ar esošo "InfoDesk" funkcionalitātes realizāciju. Iepazīšanās iterācijas lietotājstāsts ir attēlots tabulā 3.1.2.

3.1.2.tabula iepazīšanās iterācijas lietotājstāsts

Lietotāja stāsts	Uzdevums
1. Iepazīšanās ar izstrādes vidi, valodu un projektu.	1.1Iepazīties ar uzņēmuma iekšējo sistēmu
	1.2.Apgūt Eclipse izstrādes rīku un iepazīties Java valodu
	1.3.Apgūt Android izstrādes principus
	1.4.Apgūt Subversion versiju kontroles rīku
	1.5.Apgūt "InfoDesk" darbības principus un uzbūvi
	1.6.Apgūt Android Studio izstrādes vidi

3.2.2. Pirmā iterācija

Pirmās iterācijas lietotājstāsts ir attēlots tabulā 3.1.3.

3.1.3.tabula Pirmās iterācijas lietotājstāsts

Lietotāja stāsts	Uzdevums
2. Nepieciešams pārcelt prezentācijas uzstādījumus no Android ierīces	2.1.Izvērtēt iebūvētās datubāzes iespējas.
	2.2.Izveidot iespēju eksportēt konfigurācijas datus uz ārējo atmiņu un izvērtēt Google Drive iespējas
	2.3.Izveidot aktivitāti, kas pieslēgsies Google Drive.
	2.4.Izveidot aktivitāti, kas eksportē attēlus uz Google drive.
	2.5.Izveidot aktivitāti, kas eksportē XML formāta failus uz Google Drive.
	2.6.Pielāgot aktivitātes failu grupu eksportam.
	2.7.Izveidot aktivitāti, kas veido Google Drive mapes.
	2.8.Izveidot aktivitāti, kas ļauj pārlūkot Google Drive saturu no Android iekārtas.
	2.9.Pievienot funkciju, kas pārbauda failu esamību Google Drive.
	2.10.Izveidot aktivitāti, kas veic failu aizstāšanu un dzēšanu
	2.11.Apgūt Mercurial versiju kontroles rīku

3.2.3. Otrā iterācija

Otrās iterācijas lietotājstāsts ir attēlots tabulā 3.1.4.

3.1.4.tabula Otrās iterācijas lietotājstāsts

Lietotāja stāsts	Uzdevums
3. Nepieciešams pārcelt prezentācijas uzstādījumus uz citu Android ierīci	3.1.Izveidot aktivitāti, kas veic failu un failu grupu meklēšanu izmantojot vaicājumus.
	3.2.Izveidot meklēto failu esamības noteikšanas funkciju
	3.3.Izveidot aktivitāti kas veic, failu importu.
	3.4.Izveidot aktivitāti kas veic, failu grupu importu.
	3.5.Pievienot funkciju, kas prezentāciju uzstādījumus saglabā "InfoDesk" iekšējā atmiņā.

3.2.4. Trešā iterācija

Trešās iterācijas lietotājstāsts ir attēlots tabulā 3.1.5.

3.1.5.tabula Trešās iterācijas lietotājstāsts

Lietotāja stāsts	Uzdevums
4. Fona attēlu pārcelšana	4.1.Izveidot funkciju, kas nosaka fona attēlu esamību ierīcē izveidotajā prezentācijā.
	4.2.Izveidot funkciju, kas nosaka fona attēlu skaitu, nosaukumu un atrašanās vietu
	4.3.Pielāgot aktivitāti fona attēlu eksportēšanai uz Google Drive.
	4.4.Izveidot funkciju, kas nosaka fona attēlu esamību Google Drive.
	4.5.Izveidot funkciju, kas no koplietojamajiem uzstādījumiem nosaka fona attēlu pozīciju prezentācijā un šo attēlu nosaukumu
	4.6.Pielāgot aktivitāti fona attēlu importēšanai no Google Drive un nepieciešamās mapes izveidei ierīces ārējā atmiņā.
	4.7.Izveidot funkciju, kas koplietojamajos uzstādījumos izlabo fona attēlu atrašanās vietu.

3.1.5. Ceturtā iterācija

Ceturtās iterācijas lietotājstāsts ir attēlots tabulā 3.1.6.

3.1.6.tabula Ceturtās iterācijas lietotājstāsts

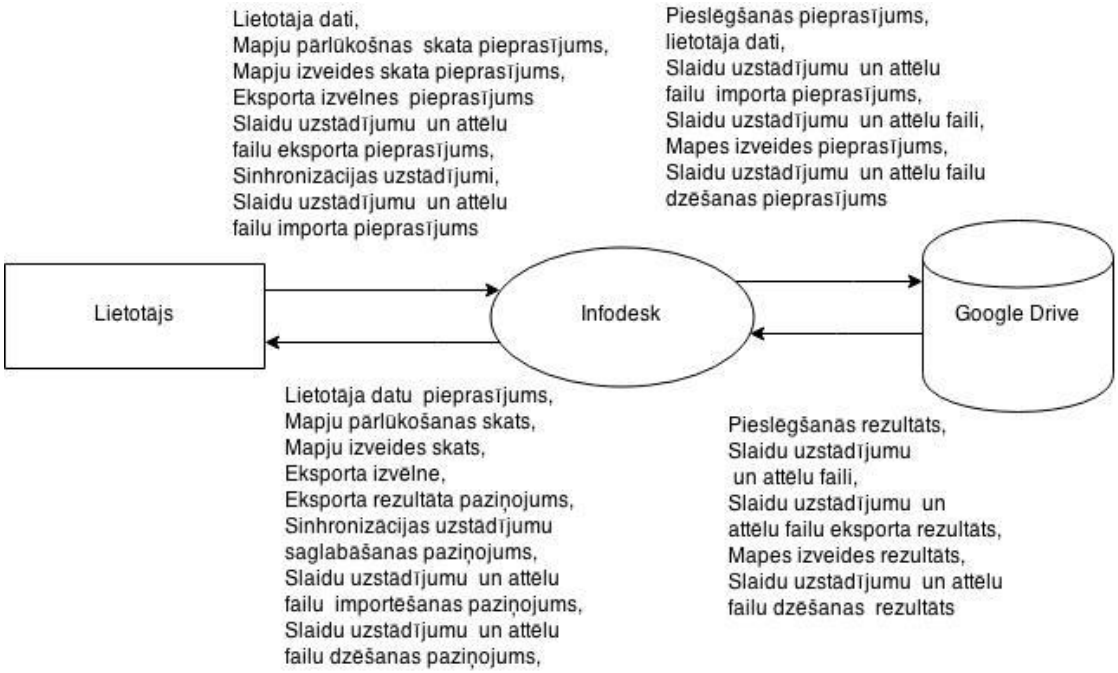
Lietotāja stāsts	Uzdevums
5. Prezentācijas uzstādījumu sinhronizācija starp ierīci un Google Drive	5.1.Izveidot aktivitāti, kas uzstādītu sinhronizācijas intervālu.
	5.2.Izveidot klasi, kas, ievērojot uzstādītu sinhronizācijas intervālu, uzsāk sinhronizācijas procesu.
	5.3.Pievienot pārbaudi interneta pieejamībai un Google Drive un ierīcē esošo uzstādījumu salīdzināšanai

4. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

Programmas projektējuma apraksta nolūks ir aprakstīt, kā lietotājstātu nodaļā minētās prasības tika realizētas programmprodukta izstrādē. Projekta sākumā nebija zināmi visi lietotājstāsti, līdz ar to arī visu lietotņu prasības, tāpēc nebija iespējams izstrādāt datu plūsmu diagrammas. Katru reizi pēc jaunu lietotājstāstu iegūšanas un to prasību noprecizēšanas, programmatūras projektējuma apraksts tika papildināts ar jaunu moduļu projektējumiem. Google drive profila piekļuvei nepieciešamā informācija turpmāk tekstā – lietotāja dati.

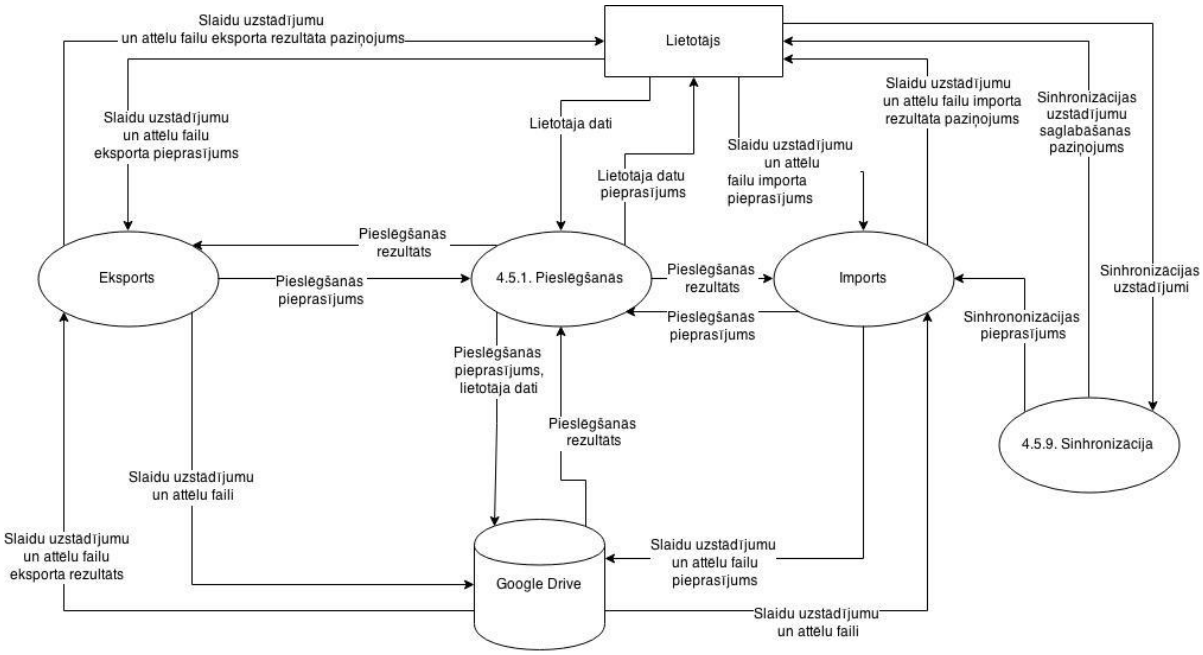
4.1. Programmatūras datu plūsmas diagrammas

"InfoDesk" tīkla funkcionalitātes 0. līmeņa datu plūsmas diagramma redzama attēlā 4.1.1



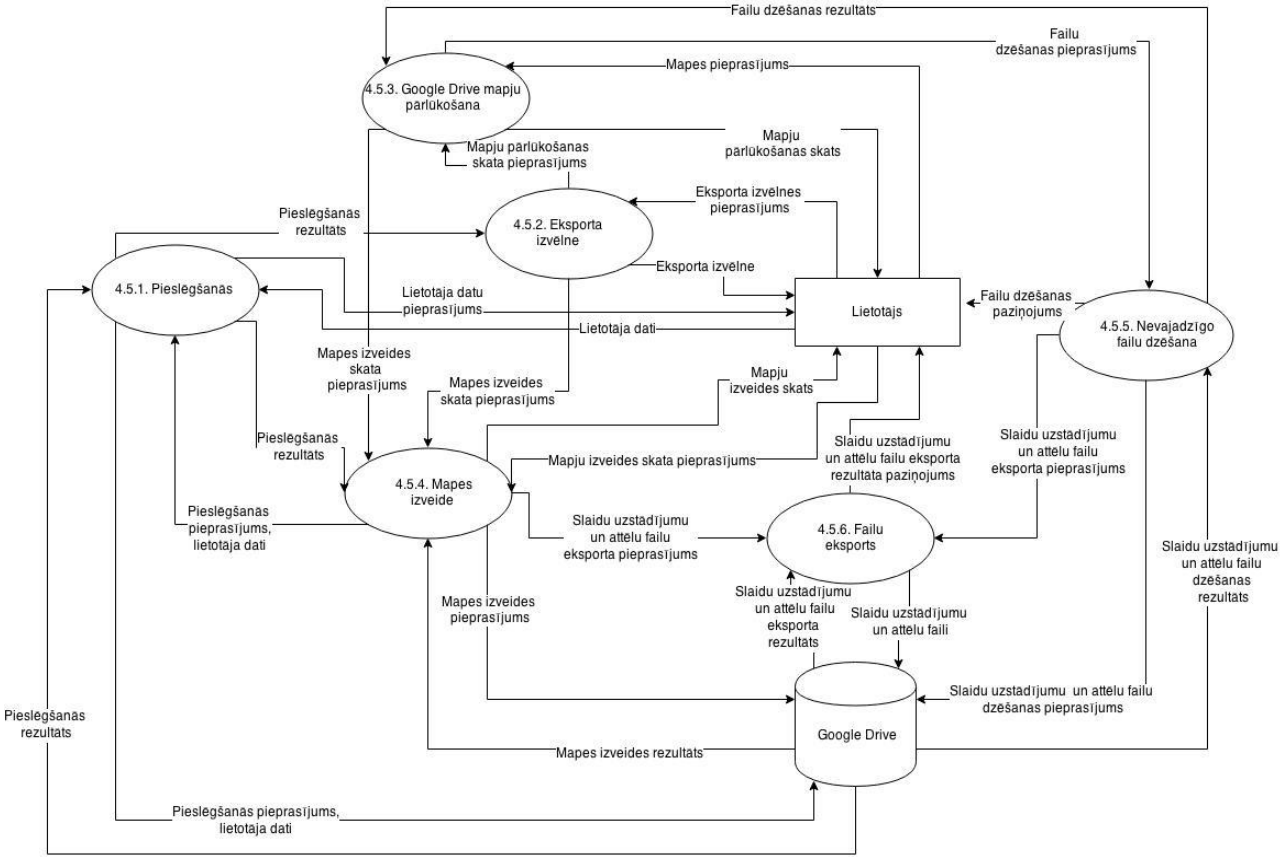
4.1.1.att. "InfoDesk" tīkla funkcionalitātes 0. līmeņa datu plūsmas diagramma

"InfoDesk" tīkla funkcionalitātes 1. līmeņa datu plūsmas diagramma redzama attēlā 4.1.2.



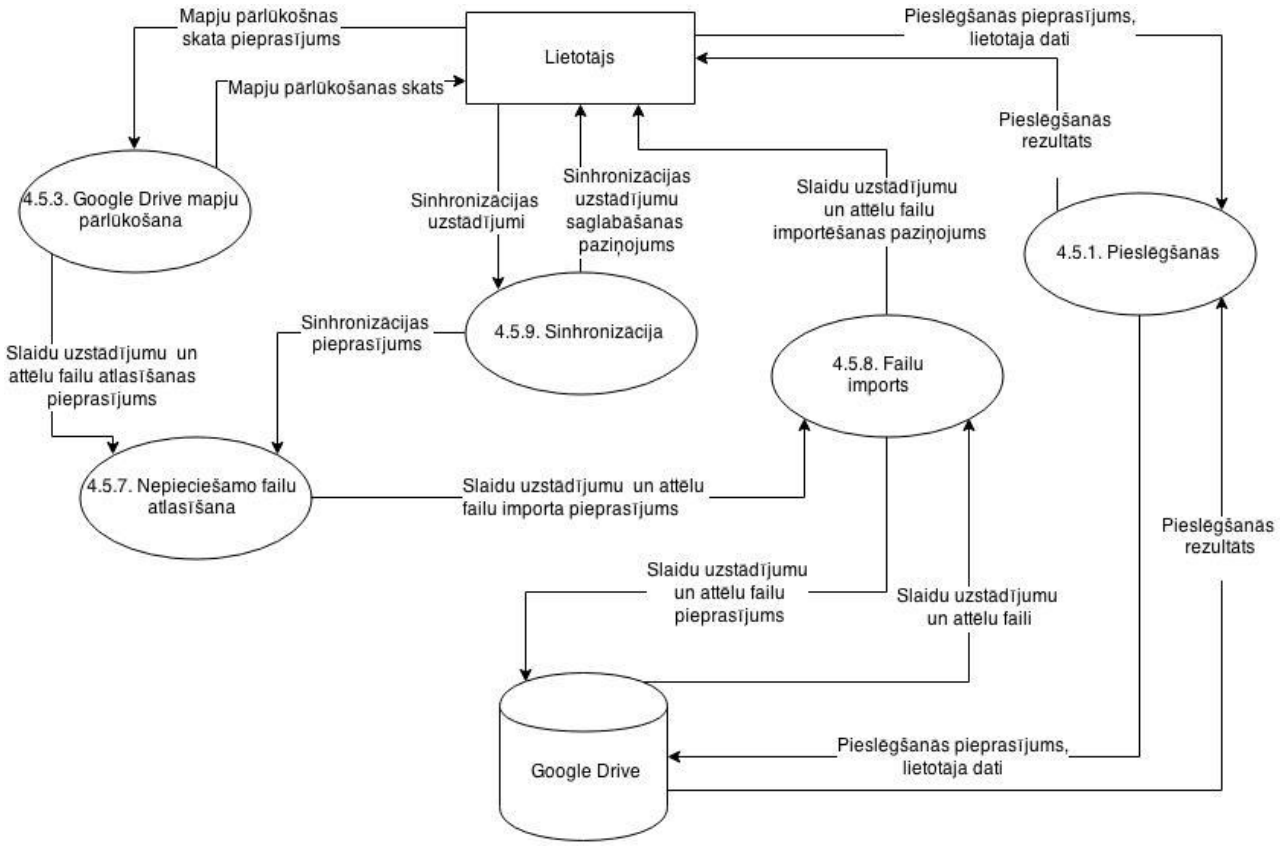
4.1.2.att. "InfoDesk" tīkla funkcionalitātes 1. līmeņa datu plūsmas diagramma

"InfoDesk" tīkla funkcionalitātes 2. līmeņa datu plūsmas diagramma, kas attēlo failu eksportu redzama attēlā 4.1.3.



4.1.3.att. "InfoDesk" tīkla funkcionalitātes 2. līmeņa datu plūsmas diagramma, kas attēlo failu eksportu

“InfoDesk” tīkla funkcionalitātes 2. līmeņa datu plūsmas diagramma, kas attēlo failu importu redzama attēlā 4.1.4.



4.1.4.att. “InfoDesk” tīkla funkcionalitātes 2. līmeņa Importa datu plūsmas diagramma

Datu plūsmu diagrammas tika izveidotas, kad bija zināmas visas lietotājstāstu prasības. Tās tika veidotas, izmantojot tiešsaistes rīku Draw.io[4]

4.2. Funkcionatlitātes piemērs

Lai izprastu lietotnes darbību, tika izveidots un apskatīts detalizēts un grafisks funkcionalitātes piemērs (*4.2.1. attēls*), kurā ir apskatīta reāla lietotāja darbība.

4.2.1. att. Funkcionalitātes piemērs prezentācijas importam eksportam un sinhronizācijai

Funkcionalitātes piemērā ir iekļauti trīs lietotājstāsti, ieslēdzot lietotni, sākumā redzama galvenā izvēlne, tālāk lietotājs importēt jau iepriekš sagatavotu prezentāciju, eksportēt paša izveidoto vai pārveidoto un uzstādīt sinhronizāciju un tās intervālu.

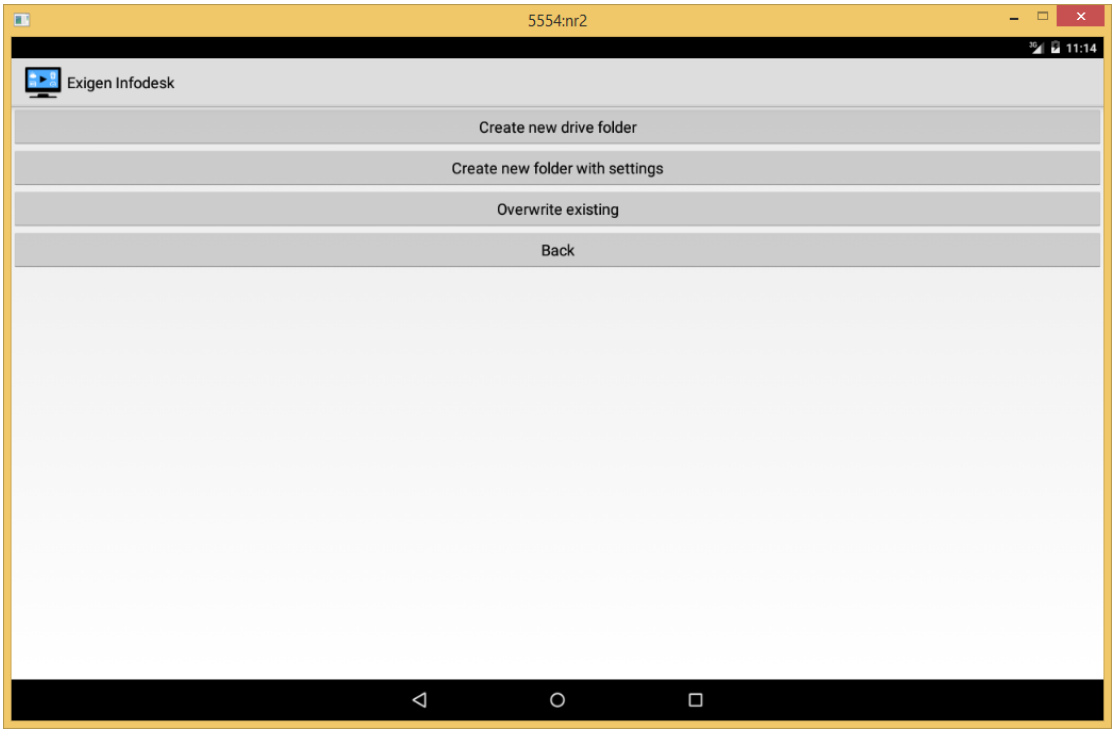
4.3. Lietotņu saskarņu diagrammas

Diagrammās ir attēlots, kā lietotne strādā noteiktu soļu gaitā.

Galvenā izvēlne tiek atvērta startējot lietotni. Galvenā izvēlne redzama attēlā 4.3.1.

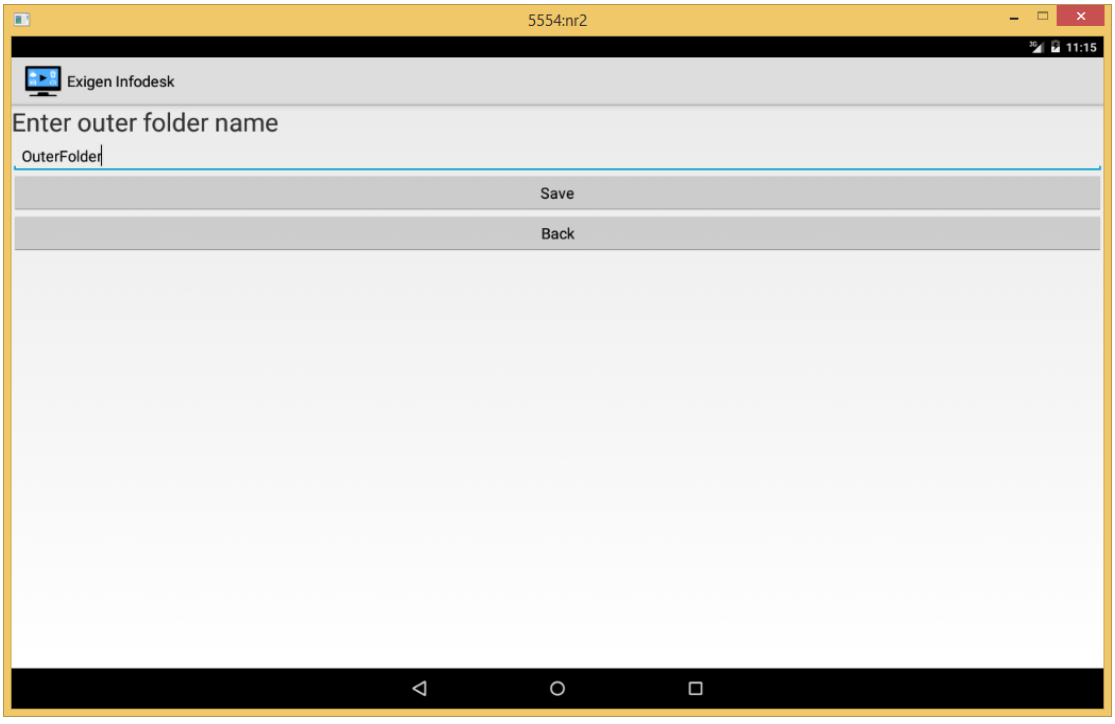
4.3.1.Att Lietotnes galvenā izvēlne

Izvēloties “Export” no galvenās izvēlnes skata lietotājs nonāk eksporta izvēlnes skatā. Eksporta izvēlne redzama attēlā 4.3.2.



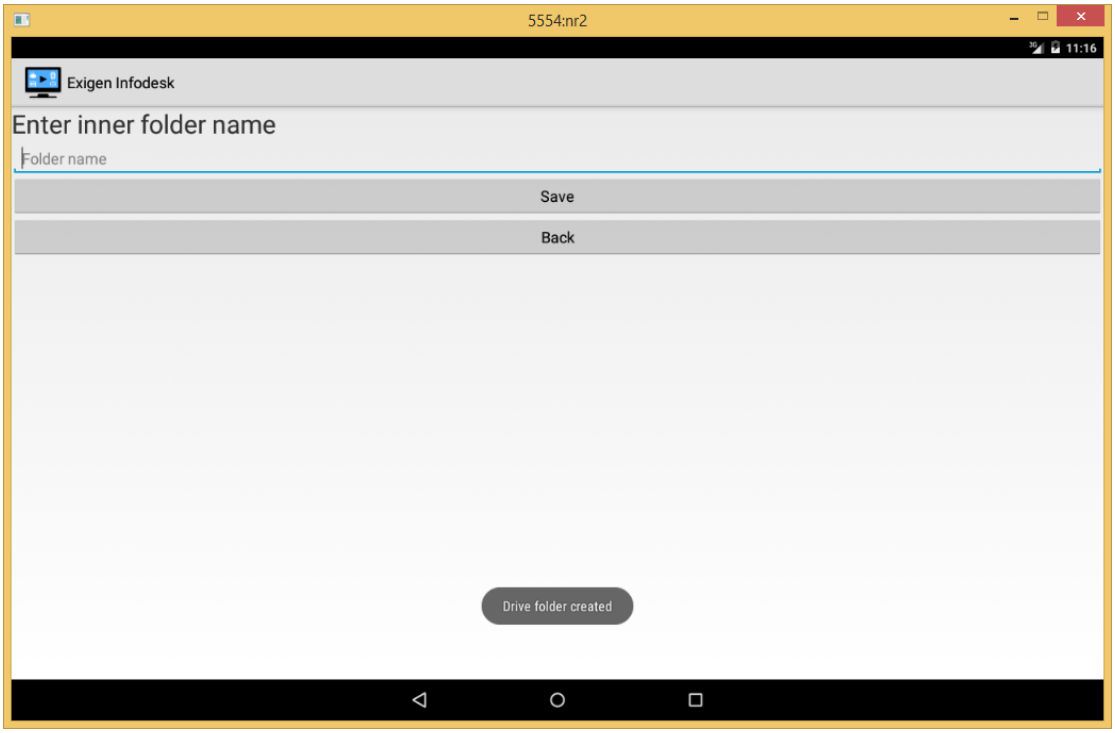
4.3.2.att Eksporta izvēlne

Izvēloties “Create new drive folder” tiek atvērta jaunas Google Drive ārējās mapes izveides skats.”Save” veic jaunas mapes izveidi, ja ir ierakstīts mapes nosaukums. “back” ir atpakaļ atgriešanās poga. Jaunas Google Drive ārējās mapes izveides skats redzams attēlā 4.3.3.



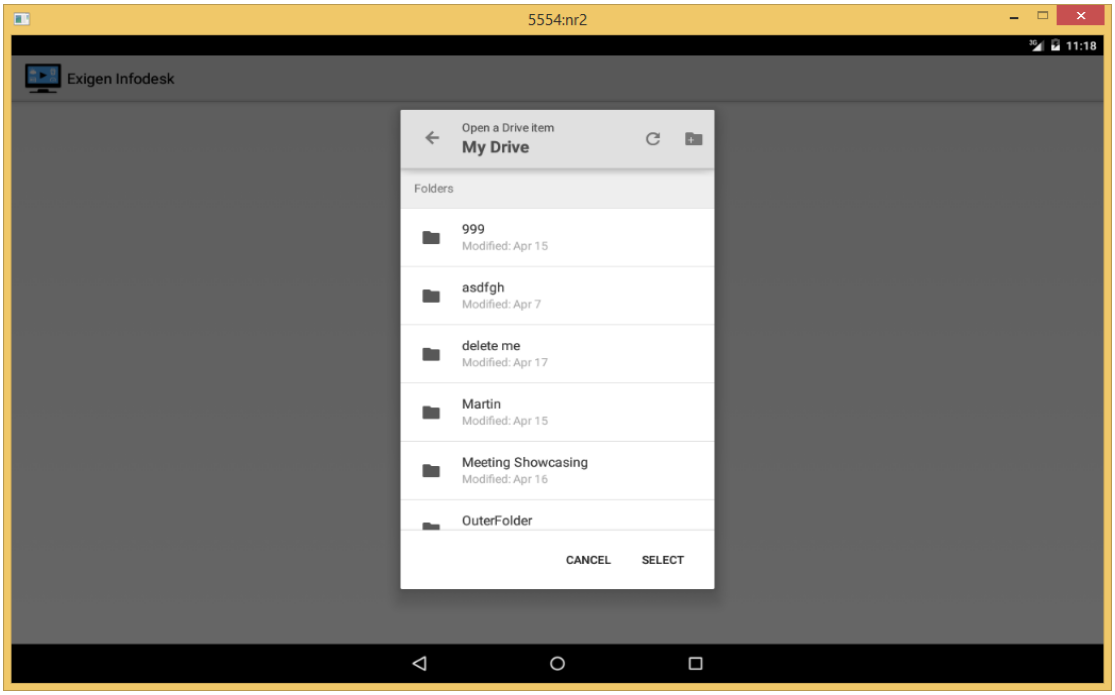
4.3.3.att Jaunas Google Drive ārējās mapes izveides skats

Izveidojot ārējo Google Drive mapi, vai Eksporta skatā izvēloties “Create new folder with settings” un pārlūkošanas skatā izvēloties mapi, tiek atvērts iekšējās Google Drive mapes izveides skats. Jaunas Google Drive iekšējās mapes izveides skats redzams attēlā 4.3.4.



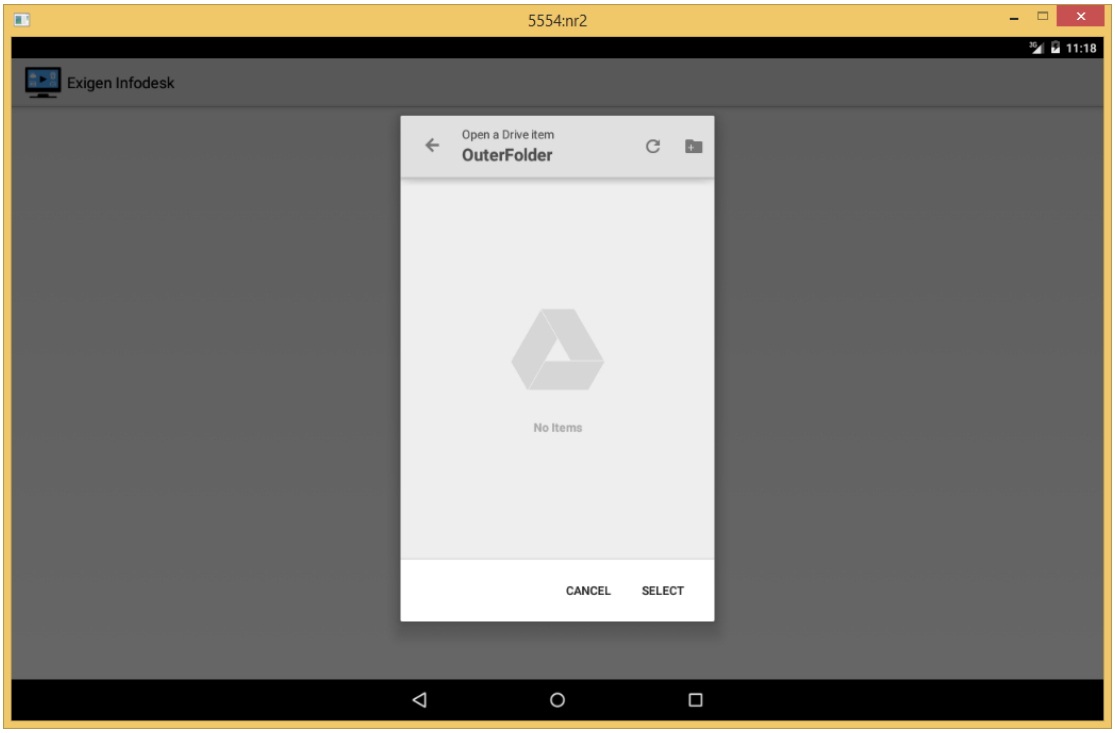
4.3.4.att Jaunas Google Drive iekšējās mapes izveides skats

Eksporta skatā izvēloties “Create new folder with settings” vai “Overwrite existing” tiek atvērts Google drive mapju pārlūkošanas skats. Google Drive mapju pārlūkošanas skats redzams attēlā 4.3.5.



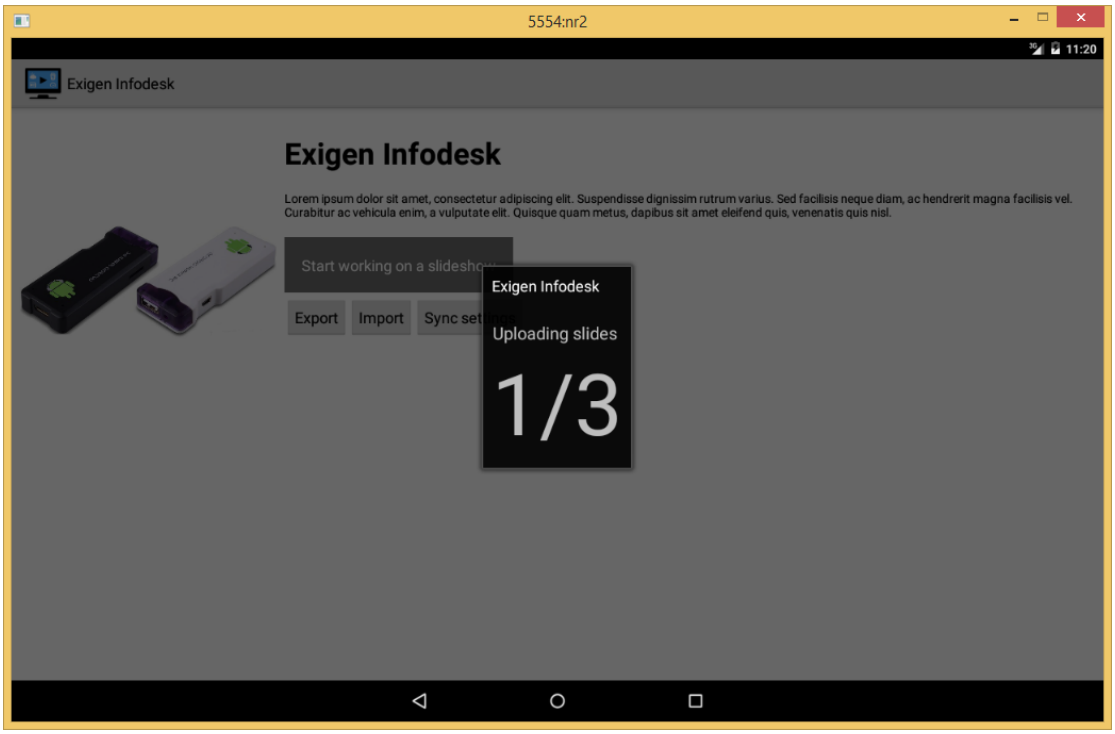
4.3.5.att Google Drive mapju pārlūkošanas skats

Atrodot nepieciešamo mapi izvēlas “select”, “cancel” pārtrauc mapju pārlūkošanu. Mapes izvēlēšanās prezentācijas eksportam redzama attēlā 4.3.6.



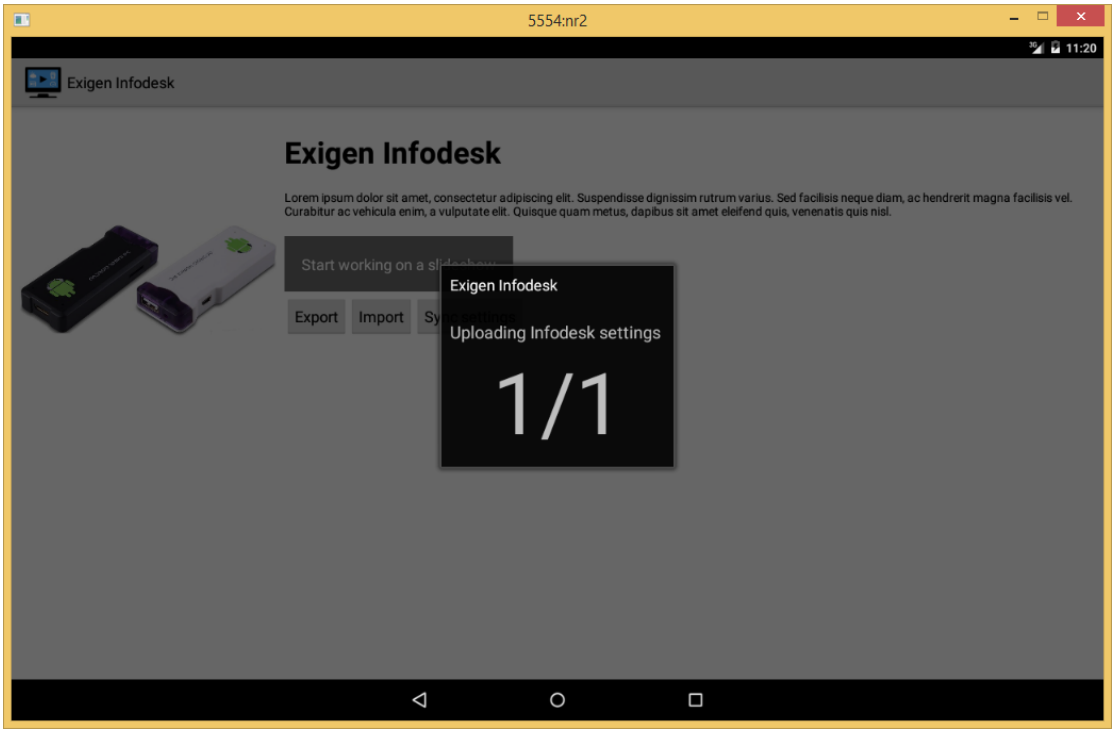
4.3.6.att Mapes izvēlēšanās prezentācijas eksportam

Izvēloties nepieciešamo mapi, tiek uzsākts failu eksports. Slaidu eksports redzams attēlā 4.3.7.



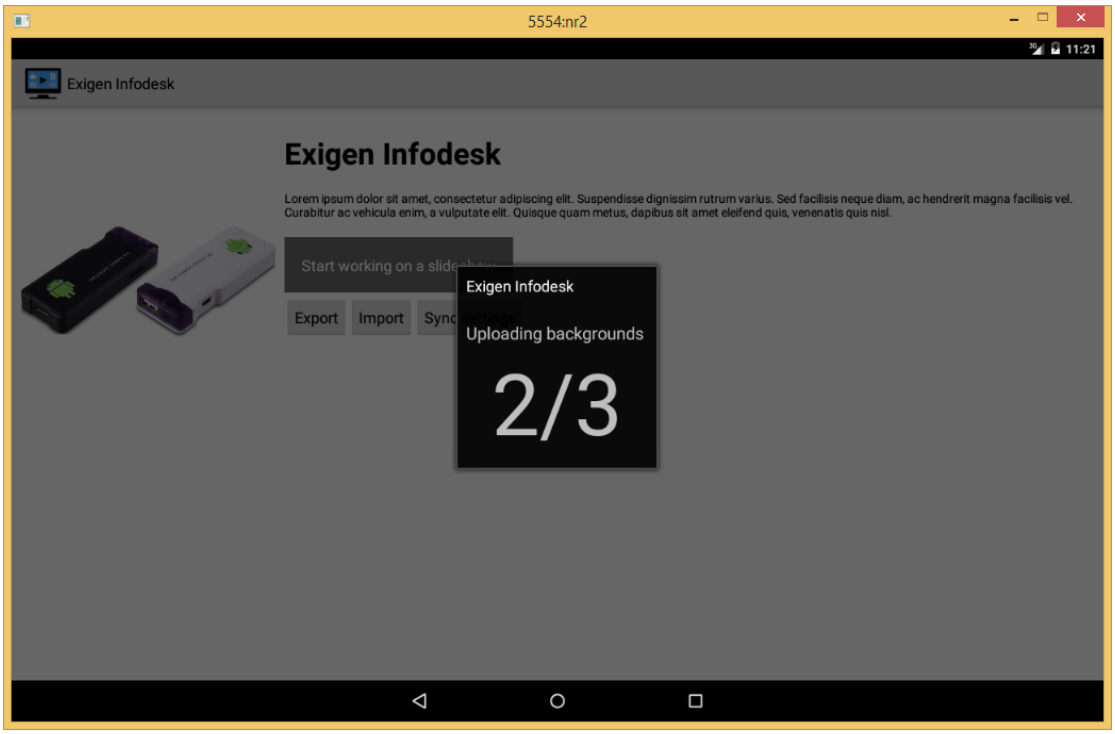
4.3.7.att Slaidu eksports

Uzstādījumu eksports redzams attēlā 4.3.8.



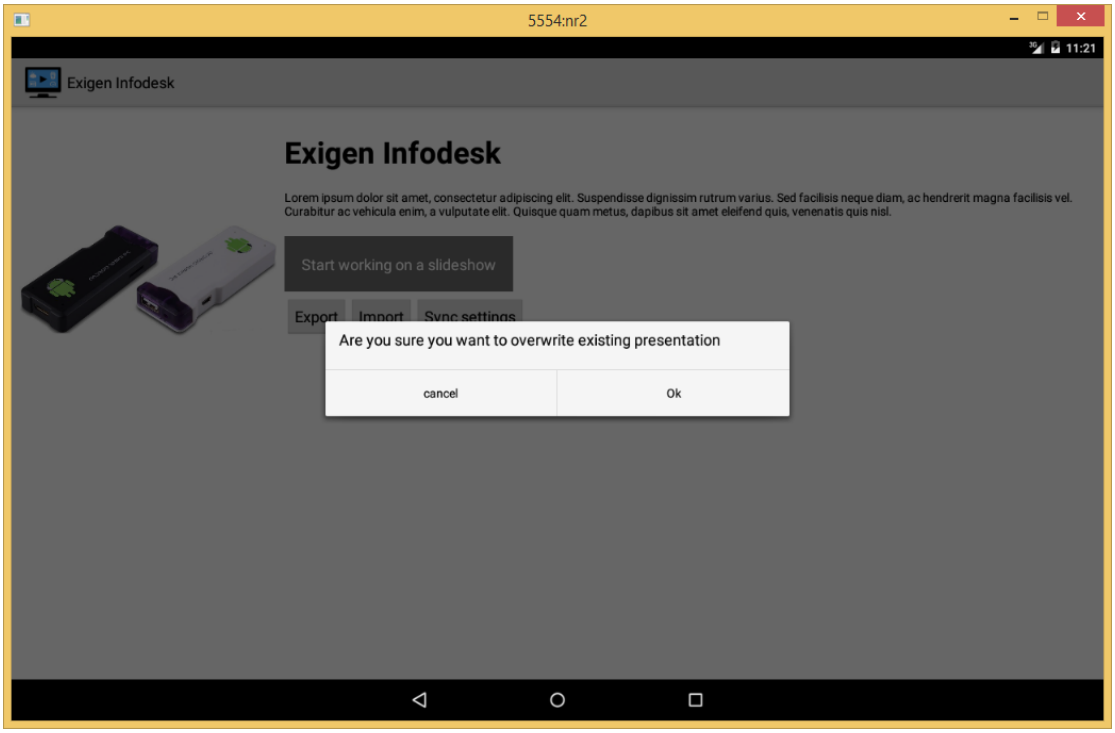
4.3.8.att Uzstādījumu eksports

Fona attēlu eksports attēlā 4.3.9.



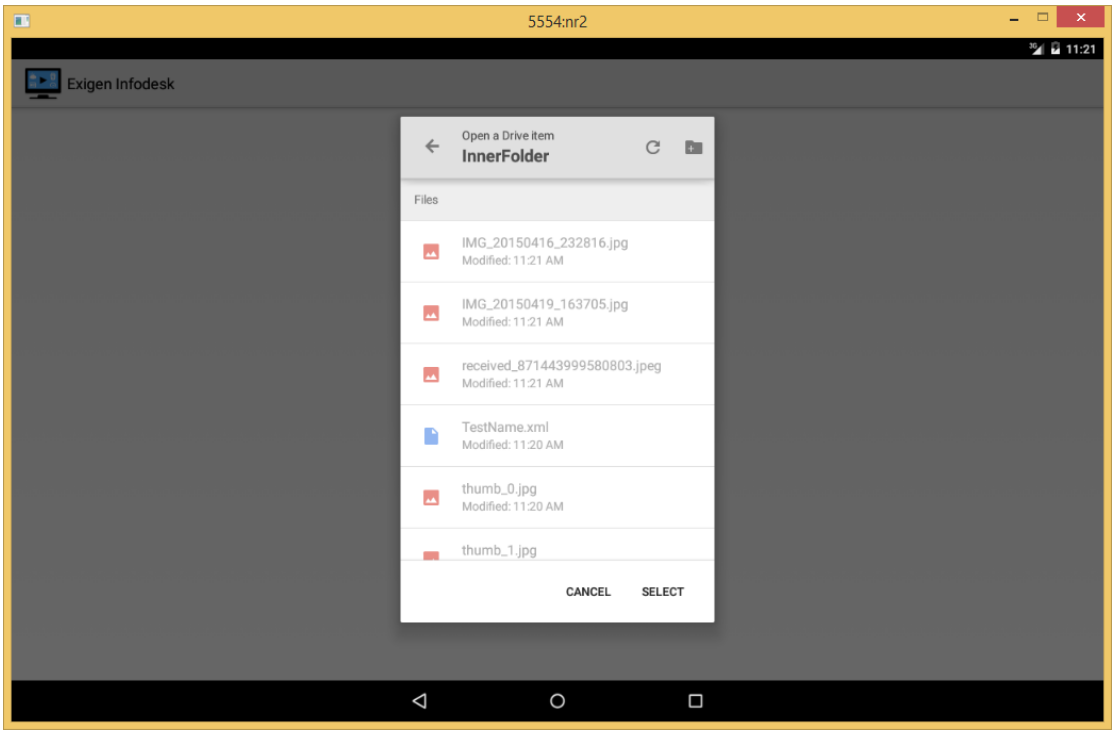
4.3.9.att Fona attēlu eksports

Izvēloties “Import” no galvenās izvēlnes skata tiek uzsākta importa process. Ja ierīcē jau atrodas iepriekš izveidota prezentācija, izvēloties “Import”, tiek parādīts kontroldialogs. Importēšanas kontroldialogs redzams attēlā 4.3.10.



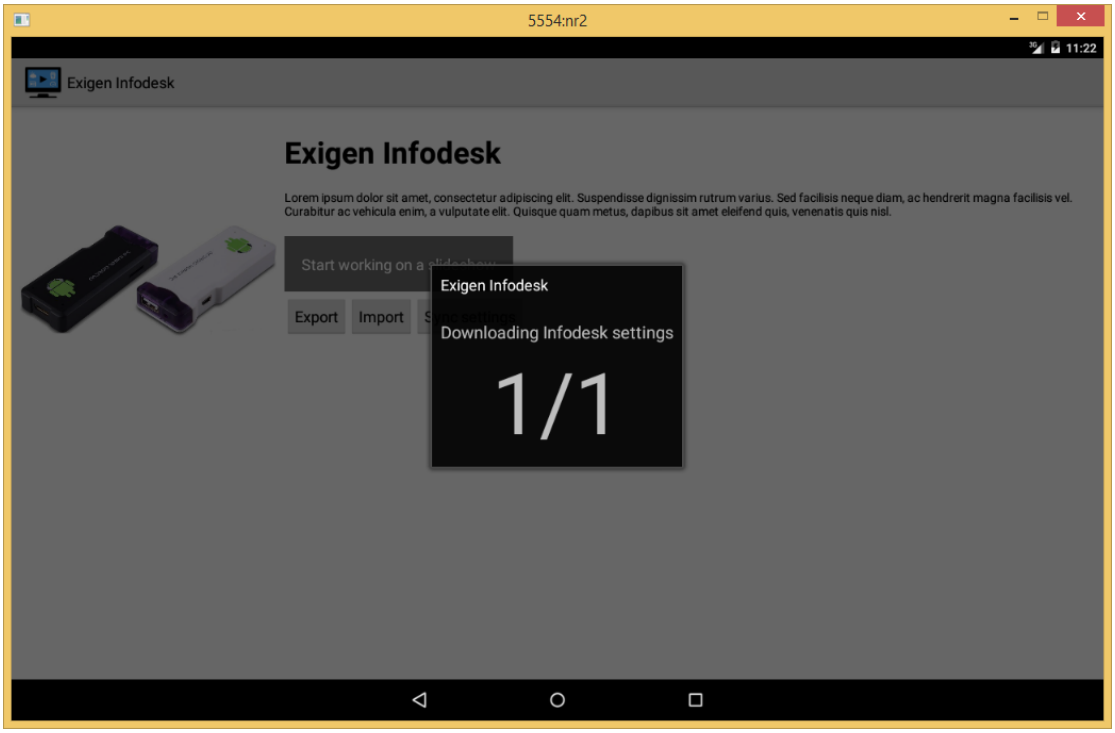
4.3.10.att Importēšanas kontroldialogs

Izvēloties “Import” no galvenās izvēlnes skata tiek atvērts Google Drive mapju pārlūkošanas skats failu importam redzama attēlā 4.3.11.



4.3.11.att Google Drive mapju pārlūkošana failu importam

Izvēloties nepieciešamo mapi tiek uzsākts failu imports. Uzstādījumu faila lejupielāde redzama attēlā 4.3.12.

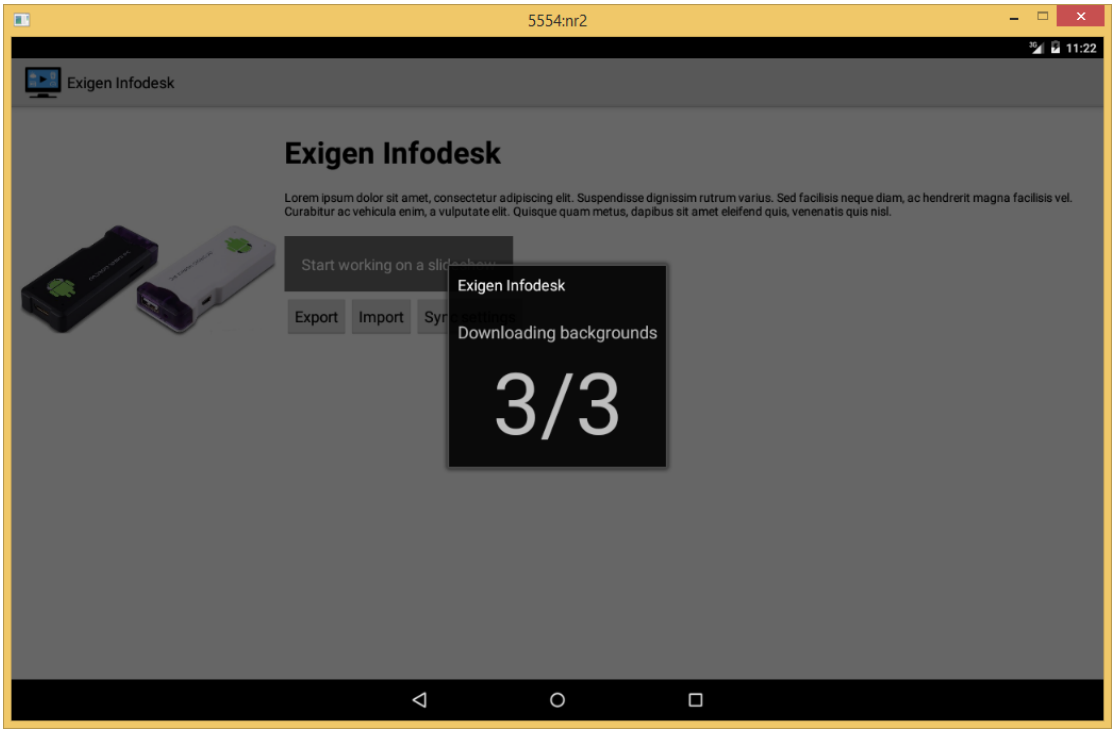


4.3.12.att Uzstādījumu faila lejupielāde

Slaidu imports redzams attēlā 4.3.13.

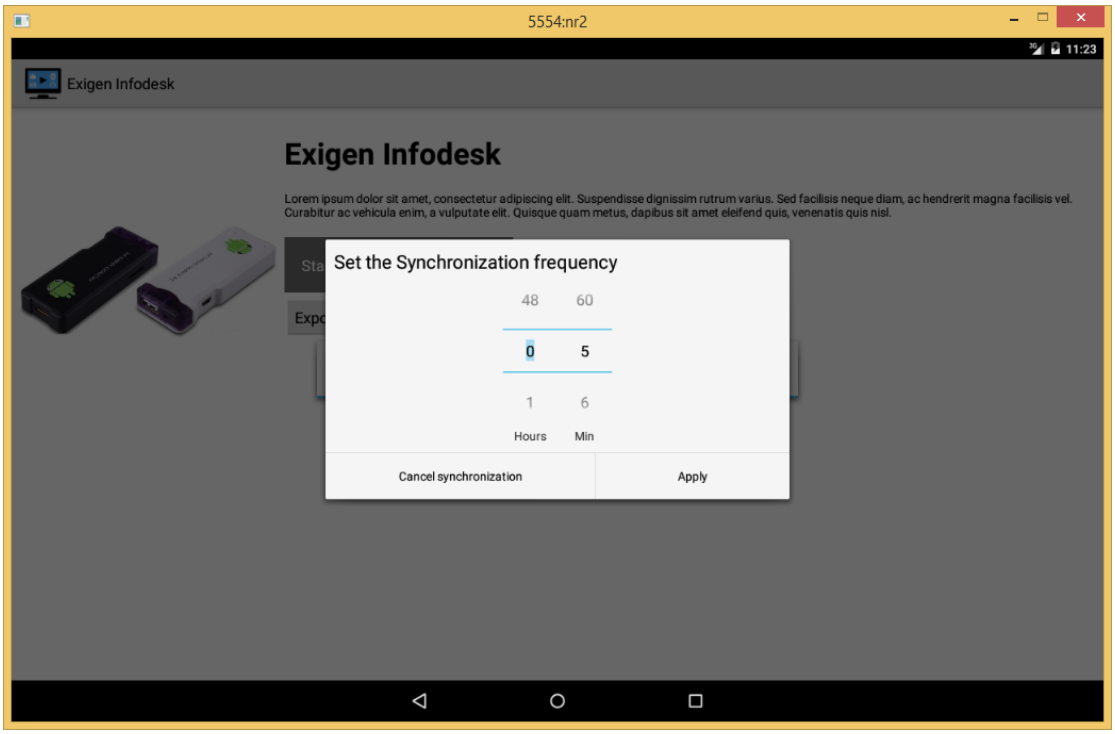
4.3.13.att Slaidu imports redzams

Fona attēlu imports redzams attēlā 4.3.14.



4.3.14.att Fona attēlu imorts

Izvēloties "Sync settings" no galvenās izvēlnes skata tiek atvērts sinhronizājas uzstādījumu dialogs. Sinhronizācijas intervāla uzstādīšana redzama attēlā 4.3.15.

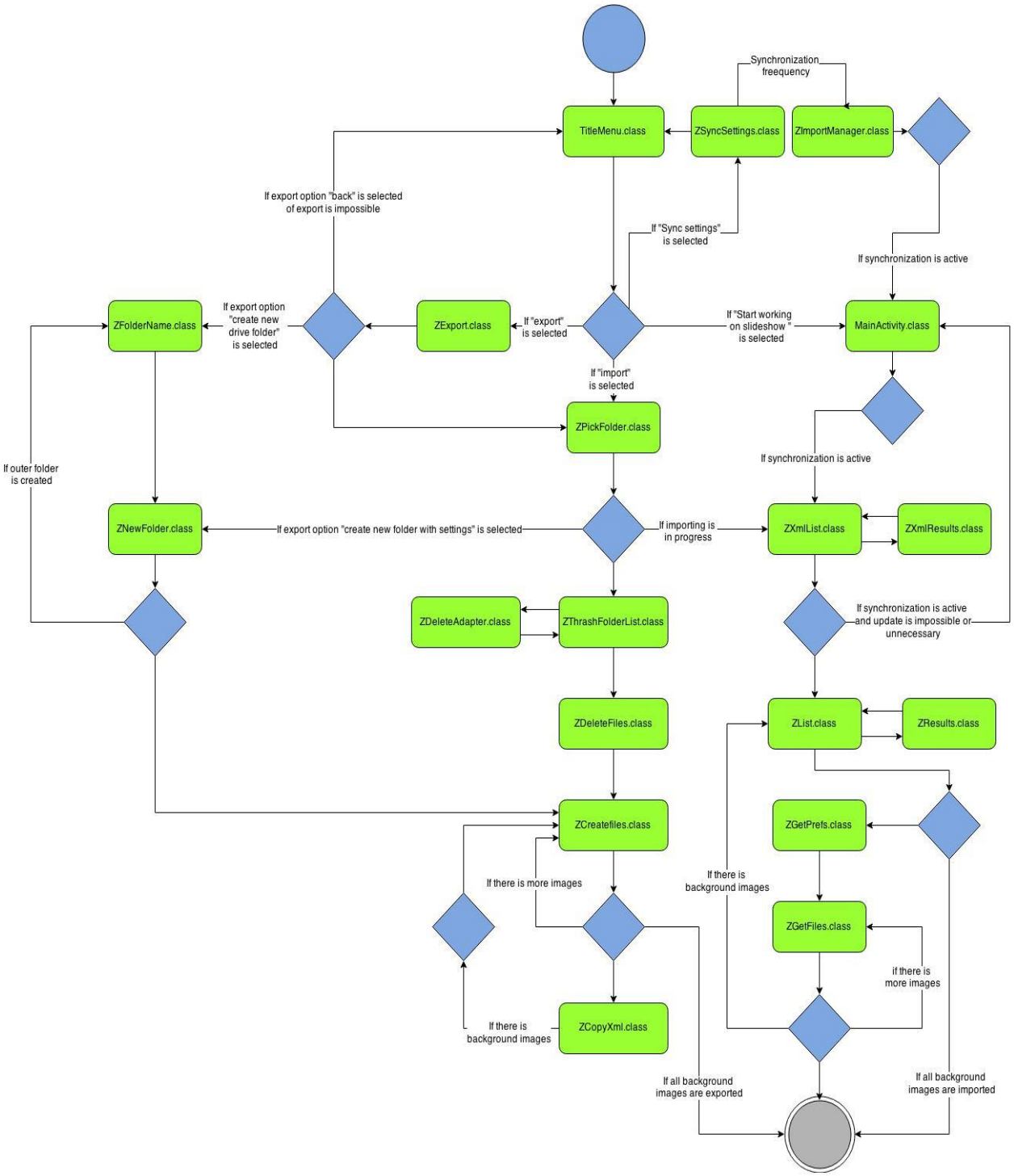


4.3.15.att Sinhronizācijas intervāla uzstādīšana

4.4. UML diagrammas

Lai izstrādātājs labāk izprastu to, kā tīkla funkcionalitāti veidojošās klases ir, saistītas savā starpā, tika izstrādāta "InfoDesk" tīkla funkcionalitātes klašu UML diagramma. Klašu metodes sīkāk aprakstītas nākamjā nodaļā

UML diagramma veidota izmantojot Draw.io. [4] "InfoDesk" tīkla funkcionalitātes klašu UML diagramma redzama attēlā 4.4.1.



4.4.1.att redzama "InfoDesk" tīkla funkcionalitātes klašu UML diagramma

4.5. Lietotnes moduļu projektējums

4.5.1. Pieslēgšanās

Klase ZBaseActivity

Klase, kura ir atbildīga par lietotnes pievienošanos Google Drive.

Metodes:

Tabula 4.5.1.1. ZBaseActivity metodes

Tips	Metode	Apraksts
void	onResume()	Veic pievienošanos Google Drive profilam
void	onPause()	Veic atvienošanos no Google Drive profila
GoogleApiClient	getGoogleApiClient()	Atgriež Google Drive profila mainīgo
void	showMessage(String message)	Parāda paziņojumu

Mainīgie:

Tabula 4.5.1.2. ZbaseActivity mainīgie

Tips	Mainīgais	Apraksts
int	REQUEST_CODE_RESOLUTION	Pieprasījuma kods
GoogleApiClient	mGoogleApiClient	Google drive profilu reprezentējošs mainīgais

Klase ZAsyncTask

Klase, kura ir atbildīga par lietotnes pievienošanos Google Drive.

Metodes:

Tabula 4.5.1.3. ZAsyncTask metodes:

Tips	Metode	Apraksts
abstract	ZAsyncTask (Context context)	Veic pievienošanos Google Drive profilam

Mainīgie:

Tabula 4.5.1.4. ZAsyncTask mainīgie

Tips	Mainīgais	Apraksts
GoogleApiClient	mClient	Google drive profilu reprezentējošs mainīgais

4.5.2. Eksporta izvēlne

Klase ZExport

Klase izveido eksporta izvēlnes skatu, attēlojot eksporta iespējas, izvēloties kādu no tām, tiek izvēlētā darbība un izsaukta attiecīgā klase.

Metodes:

Tabula 4.5.2.1. ZExport metodes

Tips	Metode	Apraksts
void	onCreate(Bundle savedInstanceState)	Izveido eksporta izvēlnes skatu un nosaka, vai ierīce satur eksportam sagatavotu prezentāciju.
void	existing(View v)	Atver izvēlētā profila Google Drive pārlūkošanas skatu, nododot intentu ZPickFolder aktivitātei.
void	back (View v)	Atgriežas uz TittleMenu klasi.
void	onClick(View v)	Nododot intentu ZPickFolder aktivitātei.
void	firstTime(View v)	Nodod intentu ZFolderName aktivitātei.

Mainīgie:

Tabula 4.5.2.2. ZExport mainīgie

Tips	Mainīgais	Apraksts
String	sharedPreferencesInternalPath	Lietotnes iekšējo failu atrašanās vieta.
PackageManager	packageManager	Ierīces pakešu pārvaldnieks
PackageInfo	package_info	Ierīces paketes informācija
String	myPath	Lietotnes iekšējās failu atrašanaš vieta.
String	myPath2	Koplietojamo uzstādījumu atrašānās vieta
File	testPrefsFile	Koplietojamo uzstādījumu fails
File	sourceLocation	"InfoDesk" slaidu attēlu atrašanās vieta
Intent	intent	Signāls saziņai starp aktivitātēm
SharedPreferences	mainPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais

4.5.3. Google Drive mapju pārlūkošana

Klase ZPickFolder

Klase, kas izsauc Google Drive pārlūkošanas skatu, un iegūst izvēlētās mapes Google Drive Id, Lietotnes tālākām darbībām.

Metodes:

Tabula 4.5.3.1. ZPickFolder metodes

Tips	Metode	Apraksts
void	onConnected(Bundle connectionHint)	Atver izvēlētā profila Google Drive pārlūkošanas skatu.
void	onActivityResult(int requestCode, int resultCode, Intent data)	Saglabā izvēlētās mapes Google Drive ID un nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.3.2. ZPickFolder mainīgie

Tips	Mainīgais	Apraksts
int	REQUEST_CODE_OPENER	Pieprasījuma kods
IntentSender	intentSender	Signāls saziņai ar Google Drive
int	requestCode	Pieprasījuma kods
int	resultCode	Rezultāta kods
Intent	data	Signāls iekšējai saziņai
DriveId	driveId	Google Drive satura elementa Id
Intent	intent	Signāls saziņai starp aktivitātēm
SharedPreferences	myPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais
SharedPreferences.Editor	e	Koplietojamo uzstādījumu izmaiņu pārvaldnieks (editors)

4.5.4. Mapes izveide

Klase ZFolderName

Klase mapes nosaukuma izveidei.

Metodes:

Tabula 4.5.4.1. ZFolderName metodes

Tips	Metode	Apraksts
void	back (View v)	Atgriežas uz TittleMenu klasi.
void	onClick(View v)	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta un saglabā mapes nosaukumu.

Mainīgie:

Tabula 4.5.4.2 ZFolderName Mainīgie

Tips	Mainīgais	Apraksts
EditText	edit	Teksta lauks, kurā lietotājs var veikt izmaiņas
TextView	txt	Teksta lauks
Intent	intent	Signāls saziņai starp aktivitātēm
SharedPreferences	myPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais
SharedPreferences.Editor	e	Koplietojamo uzstādījumu izmaiņu pārvaldnieks (editors)

Klase ZNewFolder

Klase Google Drive mapes izveidei.

Metodes:

Tabula 4.5.4.3. ZNewFolder metodes

Tips	Metode	Apraksts
void	onConnected(Bundle connectionHint)	Izveido jaunu Google Drive mapi
void	Drive.DriveApi.getFolder(GoogleApiClient, DriveId)	Izveido Google Drive mapes veidni.
void	onResult(DriveFolder.DriveFolderResult result)	Pēc Google Drive mapes izveides nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.4.4. ZNewFolder mainīgie

Tips	Mainīgais	Apraksts
SharedPreferences	myPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais
SharedPreferences.Editor	e	Koplietojamo uzstādījumu izmaiņu pārvaldnieks (editors)
MetadataChangeSet	changeSetFolder	Google Drive mapes veidne.
Intent	intent	Signāls saziņai starp aktivitātēm

4.5.5. Nevajadzīgo failu dzēšana

Klase ZTrashFolderList

Klase kas veic izdzēšanai paredzēto Google Drive failu atlasīšanu.

Metodes:

Tabula 4.5.5.1. ZTrashFolderList metodes

Tips	Metode	Apraksts
void	onCreate(Bundle b)	Izveido skatu, reģistrē rezultātu apstrādes klasi
void	retrieveNextPage()	Veic nepieciešamo failu meklēšanu
void	onResult(DriveApi.MetadataBufferResult result)	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.5.2. ZTrashFolderList mainīgie

Tips	Mainīgais	Apraksts
ListView	mListView	Saraksta skats
DataBufferAdapter<Metadata>	mResultsAdapter	Rezultātu apstrādes klases reprezentācija
String	mNextPageToken	Nākamā elementa reprezentācija
boolean	mHasMore	Norāde uz elementu esamību

Klase ZDeleteAdapter

Klase kas veic izdzēšanai paredzēto Google Drive failu atlasīšanu.

Metodes:

Tabula 4.5.5.3. ZDeleteAdapter metodes

Tips	Metode	Apraksts
view	getView(int position, View convertView, ViewGroup parent)	Saglabā iepriekš atlasīto failu Google Drive ID

Mainīgie:

Tabula 4.5.5.4. ZDeleteAdapter mainīgie

Tips	Mainīgais	Apraksts
Metadata	metadata	Faila metadati

Klase ZDeleteFiles

Klase atlasīto failu izdzēšanai.

Metodes:

Tabula 4.5.5.5. ZDeleteFiles metodes

Tips	Metode	Apraksts
void	onConnected(Bundle connectionHint)	Pārvieto iepriekš atlasītos failus uz Google Drive atkritni

Mainīgie:

Tabula 4.5.5.6. ZDeleteFiles mainīgie

Tips	Mainīgais	Apraksts
DriveFile	driveFile	Google Drive satura elementu reprezentējošs mainīgais
TextView	textView	Teksta lauks
Intent	intent	Signāls saziņai starp aktivitātēm

4.5.6. Failu eksports

Klase ZCreateFilesActivity

Klas, kas Attēlu failu eksportē uz Google Drive.

Metodes:

Tabula 4.5.6.1. ZCreateFilesActivity metodes

Tips	Metode	Apraksts
void	onConnected(Bundle connectionHint)	Izveido skatu, atrod eksportējamo failu.
ResultCallback<DriveContentsResult>	ResultCallback<DriveContentsResult>	Pievienojas Google Drive un eksportē failu.
ResultCallback<DriveFileResult>	ResultCallback<DriveFileResult>	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.6.2. ZCreateFilesActivity mainīgie

Tips	Mainīgais	Apraksts
String[]	myList	Saraksts, kas satur failu atrašanās vietu
TextView	edit	Teksta lauks
TextView	task	Teksta lauks
Integer	count	Failus skaits
Integer	skaits	Failus skaits
Bitmap	thumbnail	Attēla failu reprezentējošs mainīgais
String	thumbnailName	Attēla nosaukums
BitmapFactory.Options bmOptions	bmOptions	Attēla failu reprezentējošā mainīgā uzstādījumi
Integer	h	Nepieciešamais augstums
Integer	w	Nepieciešamais platums
OutputStream	outputStream1	Datu izvades plūsma
ByteArrayOutputStream	bitmapStream	Datu izvades plūsma bitu veidā
MetadataChangeSet	changeSet	Google Drive faila veidne.
SharedPreferences	myPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais
Intent	intent	Signāls saziņai starp aktivitātēm

Klase ZCopyXml

Klase, kas eksportē xml failu uz Google Drive.

Metodes:

Tabula 4.5.6.3. ZCopyXml metodes

Tips	Metode	Apraksts
void	onConnected(Bundle connectionHint)	Izveido skatu, pieslēdzas Google Drive
ResultCallback<DriveApi.DriveContentsResult>	ResultCallback<DriveApi.DriveContentsResult>	Eksportē failu uz failu Google Drive
ResultCallback<DriveFolder.DriveFileResult>	ResultCallback<DriveFolder.DriveFileResult>	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.6.4. ZCopyXml mainīgie

Tips	Mainīgais	Apraksts
TextView	edit	Teksta lauks
TextView	task	Teksta lauks
boolean	mainPrefs	Mainīgais, kas nosaka kāda tipa xml failu eksportēt failu
String	sharedPreferencesInternalPath	Lietotnes iekšējās failu atrašanās vieta.
PackageManager	packageManager	Ierīces pakešu pārvaldnieks
PackageInfo	package_info	Ierīces paketes informācija
File	internalFile	Failu reprezentējošais mainīgais
InputStream	inputStream	Datu ievbades plūsma
ByteArrayOutputStream	byteBuffer	Datu izvades plūsma bitu veidā
Integer	bufferSize	Datu kopas izmērs
byte[]	buffer	Bitu masīvs
byte[]	temp	Bitu masīvs
MetadataChangeSet	changeSet	Google Drive faila veidne.
SharedPreferences	mainPrefs	Galveno koplietojamo uzstādījumus reprezentējošais mainīgais

SharedPreferences	myPrefs	Koplietojamo uzstādījumus reprezentējošais mainīgais
Intent	intent	Signāls saziņai starp aktivitātēm
Integer	count	Slaidu skaits
String[]	myArray	Masīvs ar fona attēlu atrašanšās vietām.
Integer[]	myPos	Masīvs ar slaidu numuriem, kuriem ir fona attēli
Integer	backgroundCount	Fona attēlu skaits

4.5.7. Nepieciešamo failu atlasīšana

Klase ZXmlList

Klase, kas atlasa importējamo xml failu.

Metodes:

Tabula 4.5.7.1. ZXmlList metodes

Tips	Metode	Apraksts
void	onCreate(Bundle b)	Izveido skatu, reģistrē rezultātu apstrādes klasi.
void	onConnected(Bundle connectionHint)	Izsauc ResultCallback<Status> requestSync
ResultCallback<Status> requestSync	ResultCallback<Status> requestSync	Veic Google Drive pieejamā satura atjaunināšanu un ja nepieciešams izasauc retrieveNextPage()
void	retrieveNextPage()	Veic nepieciešamo failu atlasīšanu.
void	onResult(DriveApi.MetadataBufferResult result)	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.7.2. ZXmlList mainīgie

Tips	Mainīgais	Apraksts
ListView	mListView	Saraksta skats
DataBufferAdapter<Metadata>	mResultsAdapter	Rezultātu apstrādes klases reprezentācija
String	mNextPageToken	Nākamā elementa reprezentācija
boolean	mHasMore	Norāde uz elementu esamību
TextView	textView	Teksta lauks
PendingResult<Status>	sync	Google Drive satura atjaunināšanu izsaucošs signāls
SharedPreferences	myPrefs	Koplietojamos uzstādījumus reprezentējošais mainīgais
DriveId	sFolderId	Google Drive satura elementa Id
DriveFolder	folder	Google Drive mapes Id
Query	query	Failu atlasīšanas vaicājums
Intent	intent	Signāls saziņai starp aktivitātēm

Klase ZXmlResults

Klase, kas apstrādā atlasīto xml failu.

Metodes:

Tabula 4.5.7.3. ZXmlResults metodes

Tips	Metode	Apraksts
view	getView(int position, View convertView, ViewGroup parent)	Saglabā iepriekš atlasīto failu Google Drive ID

Mainīgie:

Tabula 4.5.7.4. ZXmlResults mainīgie

Tips	Metode	Apraksts
view	getView(int position, View convertView, ViewGroup parent)	Saglabā iepriekš atlasīto failu Google Drive ID
SharedPreferences	myPrefs	Koplietojamos uzstādījumus reprezentējošais mainīgais

SharedPreferences.Editor	e	Koplietojamo uzstādījumu izmaiņu pārvaldnieks (editors)
Metadata	metadata	Google Drive faila metadati

Klase ZList

Klase, kas atlasa importējamos attēlu failus failu.

Metodes:

Tabula 4.5.7.5. ZList metodes

Tips	Metode	Apraksts
void	onCreate(Bundle b)	Izveido skatu, reģistrē rezultātu apstrādes klasi, nosaka vai lejupielādējamais fails ir fona attēls un vai tas eksistē.
void	onConnected(Bundle connectionHint)	Izsauc retrieveNextPage()
void	retrieveNextPage()	Veic nepieciešamo failu atlasīšanu.
void	onResult(DriveApi.MetadataBufferResult result)	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Mainīgie:

Tabula 4.5.7.6.ZList mainīgie

Tips	Mainīgais	Apraksts
ListView	mListView	Saraksta skats
DataBufferAdapter<Metadata>	mResultsAdapter	Rezultātu apstrādes klases reprezentācija
String	mNextPageToken	Nākamā elementa reprezentācija
boolean	mHasMore	Norāde uz elementu esamību
TextView	textView	Teksta lauks
PendingResult<Status>	sync	Google Drive satura atjaunināšanu izsaucošs signāls
SharedPreferences	myPrefs	Koplietojamos uzstādījumus reprezentējošais mainīgais

DriveId	sFolderId	Google Drive satura elementa Id
DriveFolder	folder	Google Drive mapes Id
Query	query	Failu atlasīšanas vaicājums
Intent	intent	Signāls saziņai starp aktivitātēm
Integer	count	Slaidu skaits
String[]	myArray	Masīvs ar fona attēlu atrašanšās vietām.
Integer[]	myPos	Masīvs ar slaidu numuriem, kuriem ir fona attēli
Integer	backgroundCount	Fona attēlu skaits

Klase ZResults

Klase, kas apstrādā atlasītos attēlu failus.

Metodes:

Tabula 4.5.7.7. ZResults metodes

Tips	Metode	Apraksts
view	getView(int position, View convertView, ViewGroup parent)	Saglabā iepriekš atlasīto failu Google Drive ID un nosaukumu.

Mainīgie:

Tabula 4.5.7.8. *ZResults mainīgie*

Tips	Metode	Apraksts
view	getView(int position, View convertView, ViewGroup parent)	Saglabā iepriekš atlasīto failu Google Drive ID
Metadata	metadata	Google Drive faila metadati

4.5.8. Failu imports

Klase ZGetPrefs

Klase, kas apstrādā importēto xml failu.

Metodes:

Tabula 4.5.8.1. ZGetPrefs metodes

Tips	Metode	Apraksts
void	onCreate(Bundle b)	Izveido skatu, nosaka to vai izvēlētā Google Drive mapes satur prezentācijas failus, to vai izvēlētā Google Drive mapes satur to pašu prezentāciju, kas šajā sesijā tika lejupielādēta kā pēdējā, un izdzēš failus, kuri vairs nav nepieciešami.
void	open()	Pieslēdzas konkrētajam Google Drive failam
ResultCallback<DriveApi.DriveContentsResult>	ResultCallback<DriveApi.DriveContentsResult>	Aktivizē RetrieveDriveFileContents AsyncTask klasi.
void	SaveFile()	Saglabā iegūto faila saturu.

Mainīgie:

Tabula 4.5.8.2. ZGetPrefs mainīgie

Tips	Mainīgais	Apraksts
TextView	edit	Teksta lauks
TextView	task	Teksta lauks
Integer	max	Failus saturošā saraksta garums
boolean	mainPrefs	Mainīgais, kas nosaka xml faila tipu.
String	sharedPreferencesInternalPath	Lietotnes iekšējo failu atrašanās vieta.
PackageManager	packageManager	Ierīces pakešu pārvaldnieks
PackageInfo	package_info	Ierīces paketes informācija
File	internalPath	Failu reprezentējošais mainīgais
File	deleteLocation	Mapi reprezentējošais mainīgais
Integer	length	Saraksta garums

File	folder	Mapi reprezentējošais mainīgais
String	contents	Teksta faila saturs
DriveFile	file	Google Drive failu reprezentējošs mainīgais
DriveContents	driveContents	Google Drive saturu reprezentējošs mainīgais
BufferedReader	reader	Bufera ielasītājs
StringBuilder	builder	Tekstu saturošs mainīgais
String	line	Tekstu saturošs mainīgais
Intent	intent	Signāls saziņai starp aktivitātēm
String	strFileData	No Google Drive iegūts teksta faila saturs
File	innerFolder	Lietotnes iekšējo failu atrašanās vietu reprezentējošs mainīgais
File	directory	Lietotnes uzstādījumu atrašanās vietu reprezentējošs mainīgais
File	file	Failu reprezentējošais mainīgais
FileOutputStream	fOut	Datu izvades plūsma
OutputStreamWriter	osw	Datu izvades plūsmas rakstītājs

Klase RetrieveDriveFileContentsAsyncTask

Klase, kas apstrādā importējamos failus.

Metodes:

Tabula 4.5.8.3. RetrieveDriveFileContentsAsyncTask metodes

Tips	Metode	Apraksts
String	doInBackgroundConnected(DriveId... params)	Iegūst izvēlētā teksta faila saturu
void	onPostExecute(String result)	Nodod intentu nākamajai klasei, atkarībā no sākotnēji saņemtā intenta.

Klase ZGetFiles

Klase, kas apstrādā importēto attēla failu.

Metodes:

Tabula 4.5.8.4. ZGetFiles metodes

Tips	Metode	Apraksts
void	onCreate(Bundle b)	Izveido skatu.
void	onConnected(Bundle connectionHint)	Izsauc open()
void	open()	Pieslēdzas konkrētajam Google Drive failam
ResultCallback<DriveApi.DriveContentsResult>	ResultCallback<DriveApi.DriveContentsResult>	Aktivizē RetrieveDriveFileContentsAsyncTask klasi.

Mainīgie:

Tabula 4.5.8.5. *ZGetFiles mainīgie*

Tips	Mainīgais	Apraksts
TextView	edit	Teksta lauks
TextView	task	Teksta lauks
DriveId	mSelectedFileDriveId	Izvēlētais Google Drive fails
Integer	skaits	Failu saraksta garums
Integer	max	Failus saturošā saraksta garums
String	sharedPreferencesInternalPath	Lietotnes iekšējo failu atrašanaš vieta.
PackageManager	packageManager	Ierīces pakešu pārvaldnieks
PackageInfo	package_info	Ierīces paketes informācija
File	internalPath	Failu reprezentējošais mainīgais
Integer	length	Saraksta garums
ByteArrayOutputStream	contents	Datu izvades plūsma bitu veidā
DriveFile	file	Google Drive failu reprezentējošs mainīgais
DriveContents	driveContents	Google Drive saturu reprezentējošs mainīgais
Intent	intent	Signāls saziņai starp aktivitātēm
File	innerFolder	Lietotnes iekšējo failu atrašanās vietu reprezentējošs mainīgais

File	directory	Lietotnes uzstādījumu atrašanās vietu reprezentējošs mainīgais
File	file	Failu reprezentējošais mainīgais
FileOutputStream	out	Datu izvades plūsma
InputStream	inputStream	Datu ievades plūsma
Integer	bufferSize	Datu bufera izmērs
byte[]	buffer	Bitu masīvs
ByteArrayOutputStream	byteBuffer	Datu izvades plūsma bitu veidā
ByteArrayOutputStream	bitmapStream	Datu izvades plūsma bitu veidā atēlu lejupielādēšanai
String	fails	Faila nosaukums
File	innerPath	Ārējās atmiņas nesēju reprezentējošs mainīgais
File	picDirectory	Mape ārējā atmiņas nesējā
File	backPic	Fona attēlu saturošs fails
SharedPreferences	mainPrefs	Galvenos koplietojamos uzstādījumus reprezentējošais mainīgais
Integer	count	Slaidu skaits
String[]	myArray	Masīvs ar fona attēlu atrašanšās vietām.
Integer[]	myPos	Masīvs ar slaidu numuriem, kuriem ir fona attēli
Integer	backgroundCount	Fona attēlu skaits
Integer	pos	Slaida pozīcija, kurā atrodas fona attēls
SharedPreferences.Editor	ed	Galvenos koplietojamos uzstādījumu izmaiņas veicošais mainīgais

4.5.9. Sinhronizācija

Klase ZImportManager

Klase, kas nosūta signālu sinhronizācijas uzsākšanai.

Metodes:

Tabula 4.5.9.1. ZImportManager metodes

Tips	Metode	Apraksts
void	onReceive(Context context, Intent intent)	Nosūta signālu pēc noteikta intervāla
void	setAlarm(Context context)	Uzstāda apraides signāla uztveršanu
void	clearAlarm(Context context)	Atceļ apraides signāla uztveršanu

Mainīgie:

Tabula 4.5.9.2. ZImportManager mainīgie

Tips	Metode	Apraksts
String	ALARM_ACTION	Apraides darbība
long	UPDATE_FREQUENCY	Signālu intervāls
String	action	Apraides darbība
AlarmManager	alarmManager	Signālu apstrādātājs
Intent	alarmIntent	Apraides signāls
PendingIntent	pIntent	Gaidošais signāls

Klase ZSyncSettings

Klase Sinhronizācijas un tās intervāla uzstādīšanai.

Metodes:

Tabula 4.5.9.3. ZSyncSettings metodes

Tips	Metode	Apraksts
void	setTime()	Uzstāda signāla intervālu

Mainīgie:

Tabula 4.5.9.4. ZSyncSettings mainīgie

Tips	Mainīgais	Apraksts
LayoutInflater	li	Grafiskās saskarnes aizpildītājs
View	promptsView	Aktivitātes skats
NumberPicker	np1	Grafiskā skaitļu izvēlne
NumberPicker	np2	Grafiskā skaitļu izvēlne
SharedPreferences	mainPrefs	Galveno koplietojamo uzstādījumu failu

		reprezentējošs mainīgais
SharedPreferences	myPrefs	Koplietojamo uzstādījumu failu reprezentējošs mainīgais
SharedPreferences.Editor	prefsEditor	Galvenos koplietojamos uzstādījumu izmaiņas veicošais mainīgais
AlertDialog.Builder	alertDialogBuilder	Dialoga veidotājs

4.5.10. Iekšējo mainīgo uzglabāšana

Klase ZSingleton

Klase kas tiek lietota dažādu mainīgo nodošanai starp citām klasēm.

Metodes:

Tabula 4.5.10.1. ZSingleton metodes

Tips	Metode	Apraksts
konstruktors	ZSingleton()	
ZSingleton	getInstance()	Atgriež Zsingleton klasi.
Integer[]	getBckPos()	Atgriež fona attēlu pozīciju.
void	setBckPos (Integer[] value)	Uzstāda fona attēlu pozīciju.
Boolean	getFromTitle()	Atgriež boolean mainīgo.
void	setFromTitle(Boolean value)	Uzstāda boolean mainīgo.
File	getSourceLocation()	Atgriež faila atrašanās vietu.
void	setSourceLocation(File value)	Uzstāda faila atrašanās vietu.
String[]	getMyBackground()	Atgriež fona attēlu atrašanās vietu.
void	setMyBackground(String[] value)	Uzstāda fona attēlu atrašanās vietu.
String[]	getList()	Atgriež failu atrašanās vietu sarakstu.
void	setList(String[] value)	Uzstāda failu atrašanās vietu sarakstu.
DriveId	getPrefsId()	Atgriež Google Drive esoša faila Id
void	setPrefsId(DriveId value)	Uzstāda Google Drive

		esoša faila Id
DriveId	getLastPrefsId()	Atgriež Google Drive esoša faila Id.
void	setLastPrefsId(DriveId value)	Uzstāda Google Drive satura Id.
DriveId	getId()	Atgriež Google Drive satura Id.
void	setId(DriveId value)	Uzstāda Google Drive satura Id.
String	getFolderName()	Atgriež mapes nosaukumu.
void	setFolderName(String value)	Uzstāda mapes nosaukumu.
String	getFolderName2()	Atgriež mapes nosaukumu.
void	setFolderName2(String value)	Uzstāda mapes nosaukumu.
String	getMyContents()	Atgriež teksta faila saturu.
void	setMyContents(String value)	Uzstāda teksta faila saturu.
Integer	getListLength()	Atgriež saraksta garumu.
void	setListLength(Integer value)	Uzstāda saraksta garumu.
ByteArrayOutputStream	getMyByteArrayOutputStream()	Atgriež faila saturu tālākai rakstīšanai.
void	setMyByteArrayOutputStream(ByteArrayOutputStream value)	Uzstāda faila saturu tālākai rakstīšanai.
boolean	getCheck()	Atgriež boolean mainīgo.
void	setCheck(boolean value)	Uzstāda boolean mainīgo.
String[]	getFolderList()	Atgriež Google Drive mapes iekšējo failu nosaukumus.
void	setFolderList(String value, int i)	Uzstāda Google Drive mapes iekšējo failu nosaukumus.
DriveId[]	getFolderListId()	Atgriež Google Drive mapes iekšējo failu Id.
void	setFolderListId(DriveId value, int i)	Uzstāda Google Drive mapes iekšējo failu Id.
Integer	getEdit()	Atgriež skaitu.

void	setEdit(Integer value)	Uzstāda skaitu.
Integer	getCount()	Atgriež skaitu.
void	setCount(Integer value)	Uzstāda skaitu.
String	getFileName()	Atgriež faila nosaukumu.
void	setFileName(String FileName)	Uzstāda faila nosaukumu.
boolean	getRestart()	Atgriež boolean mainīgo.
void	setRestart(boolean value)	Uzstāda boolean mainīgo.
boolean	getUpdate()	Atgriež boolean mainīgo.
void	setUpdate(boolean value)	Uzstāda boolean mainīgo.
boolean	getFirstTime()	Atgriež boolean mainīgo.
void	setFirstTime(boolean value)	Uzstāda boolean mainīgo.
boolean	getWidgets()	Atgriež boolean mainīgo.
void	setWidgets(boolean value)	Uzstāda boolean mainīgo.
void	setListsToNull()	Iztukšo sarakstus.

Mainīgie:

Tabula 4.5.10.2.ZSingleton mainīgie

Tips	Mainīgais	Apraksts
ZSingleton	mInstance	Klasi reprezentējošs mainīgais
Context	context	Kontekstu reprezentējošs mainīgais.
String	fileName	Faila nosaukums
Integer	count	Elementu skaits
Integer	edit	Elementu iterators
Integer	listLength	Saraksta garums
DriveId	prefsId	Koplietojamo uzstādījumu Google Drive faila Id
DriveId	lastPrefsId	Iepriešējo lejupielādēto koplietojamo uzstādījumu Google Drive faila Id
DriveId	thumbId	Slaida attēla Google Drive faila Id
boolean	check	Nosaka lejupielādējamā satura saglabāšanas vietu
boolean	fromTitle	Nosaka kā tika aktivizēta lejupielāde
boolean	update	Nosaka vai uzsākt automātisko lejupielādi
boolean	widgets	Nosaka vai prezentācija satur logrīkus

boolean	goBack	Nosaka vai atsākt prezentāciju
String	contents	Google Drive esošā teksta faila saturs
String	folderName	Google Drive esošas mapes nosaukums
String	folderName2	Mapes kas atrodas Google Drive esošā mapē, nosaukums
String[]	folderList	Masīvs, kas satur lejupielādējamo slaidu attēlu nosaukumus
String[]	list	Masīvs, kas satur eksportam paredzētās mapes saturu
String[]	background	Masīvs, kas satur lejupielādējamo fona attēlu nosaukumus
File	sourceLocation	Mapi reprezentējošs mainīgais
Integer[]	bckPos	Masīvs, kas saturs fona attēlu pozīcijas slaidos
DriveId[]	folderListId	Masīvs, kas satur lejupielādējamo slaidu attēlu Google Drive Id
ByteArrayOutputStream	stream	Datu izvades plūsma

4.5.11. Skatu izkārtojumi

activity_listfiles.xml – satur failus atlasošo aktivitāšu izkārtojumu
activity_process.xml – satur eksporta/importa aktivitāšu izkārtojumu
export.xml – satur eksporta izvēlnes aktivitātes izkārtojumu
folder_name.xml – satur mapes izveides aktivitātes izkārtojumu
sync_timeprompt.xml – satur sinhronizācijas uzstādījumu aktivitātes izkārtojumu
tittle_menu.xml – satur galvenās izvēlnes aktivitātes izkārtojumu

4.5.12. Iepriekš nodefinētās vērtības

strings.xml – satur izvadziņu un lietotnes virsrakstu tekstus

5. PROJEKTA ORGANIZĀCIJA

Ņemot vērā, ka sākotnēji nebija precīzi noteiktas programmatūras iespējas, tika izvēlēta spējā (Agile) programmēšanas metode[10]. Programmprodukta plānošanas un izstrādes procesa laikā produkts tika atrādīts iespējamajiem lietotājiem un tādējādi tika noskaidroti lietotājstāsti.

Neskatoties uz to, ka spējā izstrāde[10] neparedz, ka lietotājstāstu akcepttestēšanu veic izstrādātājs, tā kā pasūtītāja kā tāda nebija, tad testēšanu veica pats izstrādātājs, pēc jaunu funkciju pievienošanas pārbaudot lietotnes ar akcepttestiem un vienībtestiem.

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

Programmatūras izstrādē tika izmantota Android valoda un Google Play services bibliotēka[5]

5.1. Konfigurācijas pārvaldība

SVN sistēma tika izmantota, lai kontrolētu konfigurāciju, pielietojot izstrādes vidē Android Studio iebūvēto versijkontroles rīku.

Projekta gaitā tika izlemts pāriet uz Mercurial versijkontroles rīku SourceTree[9], to izmantojot, tika izveidots izstrādes zars, kurā regulāri tikai augšupielādētas jaunākās izmaiņas, un teorētiski izplatāmās versijas zars, kurā tika glabātas stabilās versijas

5.2. Kvalitātes nodrošināšana

Lai nodrošinātu kvalitāti:

- Izstrāde tika veikta pēc Objektorientētās programmēšanas principiem.
- Tika regulāri veikta lietotnes vienībtestēšana.
- Izstrādes vidē koda noformatēšanā tika izmantots Android standarts.
- Izmaiņas tika nodotas konfigurācijas pārvaldības sistēmai pēc katra funkcionalitātes uzlabojuma
- 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 [3]. Lietotājstāsti tika novērtēti ar 13 un 21 punktu sarežģītību, jo lietotājstāsti ir salīdzinoši vienkārši formulēti, taču satur daudz sīkākus uzdevumus. Kopsummā visi lietotājstāstu sarežģītība tika novērtēta 97 punktos.

Sarežģītības punktu sadalījums pa iterācijām redzams tabulā 6.1.

Tabula 6.1. Sarežģītības punktu sadalījums pa iterācijām

Iterācijas	Sarežģītības punkti
Iepazīšanās	21
Pirmā	21
Otrā	21
Trešā	21
Ceturtā	13

Pamatojoties uz individuālo pieredzi un īsu Android apmācības kursu, kurā tika izstrādāti dažādi vienkāršoti modeļi, projekta sākumā tika pieņemts izstrādes ātrums - 8 punkti nedēļā, līdz ar to visam projektam ir nepieciešamas apmēram 12 nedēļas.

Vidējais izstrādes ātrums bija 8,1 punkti nedēļā - to izstrādātājs aprēķināja projekta beigās, līdz ar to pārējā prakses laikā izstrādātājs veltīja rūpīgākai testēšanai un dokumentācijas saglabāšanai.

7. TESTĒŠANAS DOKUMENTĀCIJA

Ņemot vērā lietotnes uzbūvi, algoritmus testēšana tika veikta, izmantojot baltās kastes metodi. Testēšanas laikā tiek apskatīti visi programmētāja paredzētie, iespējamie kļūdu gadījumi. Papildinot lietotni ar jaunu funkcionalitāti, tika veikta arī visu iepriekšējo funkciju testēšana. Tā kā katra iterācija atbilst vienam lietotājstāstam, testēšana tika sadalīta sīkāk pa izstrādātāja izvirzītajiem uzdevumiem un aprakstīta ievērojot jau izmantoto numerāciju.

7.1. Pirmās iterācijas vienībtestēšana

Pirmās iterācijas vienībtestēšana redzama tabulā 7.1.1. Atbilstošais lietotājstāsts - nepieciešams eksportēt prezentācijas uzstādījumus no Android ierīces

Tabula 7.1.1. Pirmās iterācijas vienībtestēšana

Uzdevums	Gaidāmais rezultāts	Iegūtais rezultāts iterācijā Nr.			
		1(16.03)	2(06.04)	3(20.04)	4(04.05)
2.3. Pieslēgties Google Drive profilam	Tiek izveidots savienojums ar Google Drive profilu izmantojot ievadītos lietotāja datus	+	+	+	+
2.4. Pārlūkot izvēlētā profila Google Drive saturu	Tiek atvērts Google Drive satura pārlūkošanas skats	+	+	+	+
2.5. Izveidot mapi Google Drive	Izvēlētajā Google Drive profilā tiek izveidota mape	+	+	+	+
2.6. Eksportēt attēlu uz Google Drive izvēlēto profilu	Izvēlētajā Google Drive profilā tiek izveidota attēla kopija	+	+	+	+
2.7. Eksportēt attēlu uz Google Drive izvēlēto profilu	Izvēlētajā Google Drive profilā tiek izveidota xml faila kopija	+	+	+	+
2.8. Eksportēt failu grupu uz Google Drive izvēlēto profilu	Izvēlētajā Google Drive profilā tiek izveidota failu grupas kopijas	+	+	+	+
2.9. Veikt aizstājamo failu atlasīšanu no izvēlētās mapes	Tiek izveidots saraksts ar atlasītajiem failiem	+	+	+	+
2.10. Aizstājamo failu dzēšana no izvēlētā profila Google Drive	Faili tiek izdzēsti	+	+	+	+

7.2. Otrās iterācijas vienībtestēšana

Otrās iterācijas vienībtestēšana redzama tabulā 7.2.1. Atbilstošais lietotājstāsts - nepieciešams importēt prezentācijas uzstādījumus citā Android ierīcē

Tabula 7.2.1. Otrās iterācijas vienībtestēšan

Uzdevums	Gaidāmais rezultāts	Iegūtais rezultāts iterācijā Nr.			
		1	2(06.04)	3(20.04)	4(04.05)
3.1. Veikt importējamo failu atlasīšanu no izvēlētās mapes	Tiek izveidots saraksts ar atlasītajiem failiem		+	+	+
3.2. Noteikt vai ir atlasīti visi importēšanai paredzētie faili	Apstiprinošā gadījumā tiek veiksmīgi turpināta darbība, pretējā gadījumā tiek izmests kļūdas paziņojums un turpināta iepriekšējā darbība		+	+	+
3.3. Importēt izvēlēto failu	Tiek izveidota faila kopija ierīcē		+	+	+
3.4. Importēt failu grupu	Tiek izveidota failu grupas kopijas ierīcē		+	+	+
3.5. Saglabāt importētos failus lietotnes iekšējā atmiņā	Ierīces iekšējā atmiņā tiek izveidotas importēto failu kopijas		+	+	+

7.3. Trešās iterācijas vienībtestēšana

Trešās iterācijas vienībtestēšana redzama tabulā 7.3.1. Atbilstošais lietotāstāsts - fona attēlu eksports/imports

Tabula 7.3.1. Trešās iterācijas vienībtestēšana

Uzdevums	Gaidāmais rezultāts	Iegūtais rezultāts iterācijā Nr.			
		1	2	3(20.04)	4(04.05)
4.1. Veikt importējamo fona attēlu failu atlasīšanu no izvēlētās mapes	Tiek izveidots saraksts ar atlasītajiem failiem			+	+
4.2. Noteikt fona attēlu skaitu, nosaukumu un atrašanās vietu	Tiek izveidoti īslaicīgi mainīgie ar atbilstošajiem datiem			+	+
4.3. Eksportēt fona attēlus uz Google Drive izvēlēto profilu	Izvēlētajā Google Drive profilā tiek izveidota failu grupas kopijas			+	+

4.4. Noteikt vai ir atlasīti visi importēšanai paredzētie faili	Apstiprinošā gadījumā tiek veiksmīgi turpināta darbība, pretējā gadījumā tiek izmests kļūdas paziņojums un turpināta iepriekšējā darbība			+	+
4.5. No koplietojamajiem uzstādījumiem noteikt fona attēlu pozīciju prezentācijā un šo attēlu nosaukumu	Tiek izveidoti īslaicīgi mainīgie ar atbilstošajiem datiem			+	+
4.6. Importēt fona attēlus	Tiek izveidota fona attēlu failu kopijas ierīcē			+	+
4.7. Veikt nepieciešamās izmaiņas importētajā koplietojamo uzstādījumu xml failā	Tiek veiktas izmaiņas xml failā un prezentācija ir atbilstoši importēta ierīcē			+	+

7.4. Ceturtās iterācijas vienībtestēšana

Ceturtās iterācijas vienībtestēšana redzema tabulā 7.4.1. Atbilstošasi lietotājstāsts - automātiskas sinhronizācijas izveide

Tabula 7.4.1. Ceturtās iterācijas vienībtestēšana

Uzdevums	Gaidāmais rezultāts	Iegūtais rezultāts			
		1	2	3	4(04.05)
5.1. Uzstādīt sinhronizācijas intervālu	Tiek uzstādīts sinhronizācijas intervāls, kas tiek saglabāts koplietojamajos uzstādījumos.				+
5.2. Automātiska sinhronizācijas procesa uzsākšana pēc izvēlētā intervāla	Tiek uzsākts sinhronizācijas process				+
5.3.Noteikt vai ir pieejams internets	Apstiprinošā gadījumā tiek veiksmīgi turpināta darbība, pretējā gadījumā tiek izmests kļūdas paziņojums un turpināta iepriekšējā darbība				+

SECINĀJUMI

Android lietotņu izstrādē autoram pirms šī projekta izstādes nebija nekādas pieredzes, taču uzņēmums, kurā tika iziet prakse, nodrošināja iespēju iegūt īsu apmācību darbam ar Android lietotnēm. Šīs apmācības ietvaros apgūstot pamatzināšanas, lai varētu uzsākt darbu pie projekta “InfoDesk”

Izstrādājot “InfoDesk” lietotnes tīkla funkcionalitāti autors saskārās ar dažādām Google Drive integrācijas nepilnībām, ko izstrādātāji laika gaitā novērsa.

Nākotnē šai lietotnei varētu uzlabot ātrdarbību un atmiņas resursu izmantošanu, kā arī daļu no aktivitātēm izveidot kā fona procesus.

IZMANTOTĀ LITERATŪRA

1. А. Голощапов, *Google Android Создание приложений для смартфонов и планшетных ПК,* БХВ-Петербург, 2013
2. Android Developer. [tiešsaiste]. [Skatīts 04.02.2015]. Pieejams: http://developer.android.com/reference/packages.html
3. Darbietilpības prognozēšana [tiešsaiste]. [Skatīts 02.03.2015]. Pieejams: http://estudijas.lu.lv/pluginfile.php/258971/mod_resource/content/1/Darbietilpiibas%20prognozeeshana%20-%20Liva%20Steinberga%20-%2029%2010%202012.pdf
4. Draw.io. [tiešsaiste]. [Skatīts 20.05.2015]. Pieejams: https://www.draw.io
5. Google Play servisu uzstādīšana [tiešsaiste]. [Skatīts 02.03.2015] Pieejams: http://developer.android.com/google/play-services/setup.html
6. How to check internet access on Android? [tiešsaiste]. [Skatīts 24.03.2015] Pieejams: http://stackoverflow.com/questions/1560788/how-to-check-internet-access-on-android-inetaddress-never-timeouts
7. K. Upenieks, M. Aldiņš, R. Kļava, T. Lācis, T. Kirtovskis, O. Siliņš "Exigen "InfoDesk" tehniskā dokumentācija" 2015
8. SHA1 parakstīšanas sertifikāta iegūšana [tiešsaiste]. [Skatīts 02.03.2015] Pieejams: http://stackoverflow.com/questions/12214467/how-to-obtain-signing-certificate-fingerprint-sha1-for-oauth-2-0-on-android
9. SourceTree. [tiešaiste]. [Skatīts 15.04.2015]. Pieejams: https://www.sourcetreeapp.com/
10. Spējā izstrāde [tiešsaiste]. [Skatīts 10.03.2015]. Pieejams: http://estudijas.lu.lv/mod/page/view.php?id=128545

PIELIKUMS

Klase ZgetFiles. Šī klase pardzēta failu lejupielādēšanai no Google Drive. Tā ir veidota izmantojot Google izstrādātāju paraugu. Klases autors ir Toms Kirtovskis

```
package com.exigenservices.infodesk;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.widget.TextView;

import com.exigenservices.util.IoUtils;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.drive.Drive;
import com.google.android.gms.drive.DriveApi;
import com.google.android.gms.drive.DriveContents;
import com.google.android.gms.drive.DriveFile;
import com.google.android.gms.drive.DriveId;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class ZGetFiles extends ZBaseActivity {

   private Integer skaits = ZSingleton.getInstance().getListLength();

   private DriveId mSelectedFileDriveId=
ZSingleton.getInstance().getFolderListId()[ZSingleton.getInstance().getEdit()];

   private TextView task;
   private TextView edit;

   @Override
   protected void onCreate(Bundle b) {
      super.onCreate(b);
```

```
        Integer max;
        setContentView(R.layout.activity_process);
        edit = (TextView) findViewById(R.id.progress);
        task = (TextView) findViewById(R.id.task);
        //view in case of background import
        if (getIntent().getBooleanExtra("isBackground",false)) {
            max = skaits;
            edit.setText(ZSingleton.getInstance().getEdit() + 1+ "/" + max);
            task.setText("Downloading backgrounds");
        } else { //view in case of slide import
            max = skaits ;
            edit.setText(ZSingleton.getInstance().getEdit() + 1 + "/" + max);
            task.setText("Downloading slides");
        }
    }
    @Override
    public void onConnected(Bundle connectionHint) {
        super.onConnected(connectionHint);
        open();
    }
    @Override
    protected void onStop() {
      super.onStop();

    }
    @Override
    protected String doInBackgroundConnected(DriveId... params) {
        return null;
    }
    // open selected drive
    private void open() {
        DriveFile.DownloadProgressListener listener = new
DriveFile.DownloadProgressListener() {
            @Override
            public void onProgress(long bytesDownloaded, long bytesExpected) {
            }
        };
            Drive.DriveApi.getFile(getGoogleApiClient(), mSelectedFileDriveId)
                    .open(getGoogleApiClient(), DriveFile.MODE_READ_ONLY, listener)
                    .setResultCallback(driveContentsCallback);
    }
    private ResultCallback<DriveApi.DriveContentsResult> driveContentsCallback = new
ResultCallback<DriveApi.DriveContentsResult>() {
        //handles import results
        @Override
```

```
        public void onResult(DriveApi.DriveContentsResult result) {
            if (!result.getStatus().isSuccess()) {
                showMessage("Update failed " +"Error while reading from the file");
                ZSingleton.getInstance().setCheck(false);
                ZSingleton.getInstance().setEdit(0);
                ZSingleton.getInstance().setListsToNull();
                ZSingleton.getInstance().setPrefsId(null);
                //in case of import form title menu
                if (ZSingleton.getInstance().getFromTitle()) {
                    ZSingleton.getInstance().setFromTitle(false);
                    ZSingleton.getInstance().setRestart(false);
                    finish();
                } else { //in case of import form sync attempt
                    ZSingleton.getInstance().setRestart(true);
                    ZSingleton.getInstance().setFromTitle(false);
                    Intent intent = new Intent(getApplicationContext(), MainMenu.class);
                    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    intent.putExtra("isBackground", false);
                    startActivity(intent);
                    finish();
                }
                return;
            }
            new
RetrieveDriveFileContentsAsyncTask(ZGetFiles.this).execute(mSelectedFileDriveId);
        }
    };
    final private class RetrieveDriveFileContentsAsyncTask
            extends ZAsyncTask<DriveId, Boolean,String> {

        public RetrieveDriveFileContentsAsyncTask(Context context) {
            super(context);
        }
        //create outputStream from google drive contents
        @Override
        protected String doInBackgroundConnected(DriveId... params) {
            ByteArrayOutputStream contents;
            DriveFile file = Drive.DriveApi.getFile(getGoogleApiClient(), params[0]);
            DriveApi.DriveContentsResult driveContentsResult =
                    file.open(getGoogleApiClient(), DriveFile.MODE_READ_ONLY, null).await();
            if (!driveContentsResult.getStatus().isSuccess()) {
                return null;
            }
            DriveContents driveContents = driveContentsResult.getDriveContents();
            InputStream inputStream = driveContents.getInputStream();
```

```
    ByteArrayOutputStream byteBuffer = new ByteArrayOutputStream();
    int bufferSize = 1024;
    int bufferLength = 0;
    byte[] buffer = new byte[bufferSize];
    try {
        bufferLength = inputStream.read(buffer);
    } catch (IOException e) {
        showMessage("error");
    }
    while (bufferLength != -1) {
        byteBuffer.write(buffer, 0, bufferLength);
        try {
            bufferLength = inputStream.read(buffer);
        } catch (IOException e) {
            showMessage("error");
        }
    }
    contents = byteBuffer;
    driveContents.discard(getGoogleApiClient());
    ZSingleton.getInstance().setMyByteArrayOutputStream(contents);
    return "ok";
}

//handles results from getting drive content
@Override
protected void onPostExecute(String result) {
    super.onPostExecute(result);
    if (result == null) {
        showMessage("Update failed " +"Error while reading from the file");
        ZSingleton.getInstance().setCheck(false);
        ZSingleton.getInstance().setEdit(0);
        ZSingleton.getInstance().setListsToNull();
        ZSingleton.getInstance().setPrefsId(null);

        if (ZSingleton.getInstance().getFromTitle()) {
            ZSingleton.getInstance().setFromTitle(false);
            ZSingleton.getInstance().setRestart(false);
            finish();
        } else {
            ZSingleton.getInstance().setRestart(true);
            ZSingleton.getInstance().setFromTitle(false);
            Intent intent = new Intent(getApplicationContext(), MainMenu.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            intent.putExtra("isBackground", false);
            startActivity(intent);
```

```
            finish();
        }
        return;
    }
    // saves files in case of slide images
    if (!getIntent().getBooleanExtra("isBackground", false)) {
        PackageManager packageManager =
getApplicationContext().getPackageManager();
        String sharedPreferencesInternalPath =
getApplicationContext().getPackageName();
        try {
            PackageInfo p =
packageManager.getPackageInfo(sharedPreferencesInternalPath, 0);
            sharedPreferencesInternalPath = p.applicationInfo.dataDir;
        } catch (PackageManager.NameNotFoundException e) { //automatic
        }
        File innerPath = new File(sharedPreferencesInternalPath)
                ;
        File directory = new File(innerPath.getAbsolutePath() + "/files"
        );
        //Now create the file in the above directory and write the contents into it
        File file = new File(directory,ZSingleton.getInstance().getFolderList()[
ZSingleton.getInstance().getEdit()]);
        FileOutputStream out = null;
        try {
            out = new FileOutputStream(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream bitmapStream =
ZSingleton.getInstance().getMyByteArrayOutputStream();

        try {
            if (out != null) {
                out.write(bitmapStream.toByteArray());
                out.flush();
                out.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
            IoUtils.closeQuietly(out);
        }
        //sends intent to next activity
        if (ZSingleton.getInstance().getCheck()) {
            if (ZSingleton.getInstance().getEdit() < skaits - 1) {
```

```
                ZSingleton.getInstance().setEdit(ZSingleton.getInstance().getEdit() + 1);
                  mSelectedFileDriveId = null;
                  Intent intent = new Intent(getApplicationContext(),ZGetFiles.class);
                  ZSingleton.getInstance().setCheck(true);
                  intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(intent);
                  finish();
                } else {
                  Intent intent = new Intent(getApplicationContext(),ZList.class);
                  intent.putExtra("isBackground", true);
                  ZSingleton.getInstance().setCheck(false);
                  ZSingleton.getInstance().setEdit(0);
                  intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(intent);
                  finish();
                }
              } else   {
                  Intent intent = new Intent(getApplicationContext(),ZList.class);
                  intent.putExtra("isBackground", true);
                  ZSingleton.getInstance().setCheck(false);
                  ZSingleton.getInstance().setEdit(0);
                  intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(intent);
                  finish();
              }
            } // saves files in case of background images
            else  if (getIntent().getBooleanExtra("isBackground", false))
            {
                String fails =
ZSingleton.getInstance().getFolderList()[ZSingleton.getInstance().getEdit()];
                //check for right list
                if(getIntent().getBooleanExtra("isBackground", false) &&
fails.toLowerCase().contains("thumb")) {
                  ZSingleton.getInstance().setCheck(false);
                  ZSingleton.getInstance().setEdit(0);
                  ZSingleton.getInstance().setListsToNull();
                  ZSingleton.getInstance().setPrefsId(null);
                  if (ZSingleton.getInstance().getFromTitle()) {
                    ZSingleton.getInstance().setFromTitle(false);
                    ZSingleton.getInstance().setRestart(false);
                    finish();
                  } else {
                    ZSingleton.getInstance().setRestart(true);
                    ZSingleton.getInstance().setFromTitle(false);
                    Intent intent = new Intent(getApplicationContext(), MainMenu.class);
```

```
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                intent.putExtra("isBackground", false);
                startActivity(intent);
                finish();
             }
          } else  {
             File innerPath = Environment.getExternalStorageDirectory();
             File picDirectory = new File(innerPath.getAbsolutePath() +
"/Download/Infodesk");
             if (!picDirectory.exists()) {
                picDirectory.mkdir();
             }
             File backPic = new
File(picDirectory+"/"+ZSingleton.getInstance().getFolderList()[ZSingleton.getInstance().getE
dit()]);
            SharedPreferences mainPrefs = getApplicationContext()

.getSharedPreferences(MainActivity.SHARED_PREFERENCES_NAME,Context.MODE_M
ULTI_PROCESS );
             SharedPreferences.Editor ed = mainPrefs.edit();
             if (backPic.exists()) {
                if (ZSingleton.getInstance().getCount() > 0) {
                   Integer pos;
                   pos = ZSingleton.getInstance().getCount() - 1 -
ZSingleton.getInstance().getEdit();
                   if (pos >= 0) {

                      ed.putString("Backgroundurl" +
ZSingleton.getInstance().getBckPos()[pos]
                            , backPic.getAbsolutePath());
                      ed.commit();
                   }
                }

             } else if (!backPic.exists() ) {
                File file = new File(picDirectory,

ZSingleton.getInstance().getFolderList()[ZSingleton.getInstance().getEdit()]

                );
                FileOutputStream out = null;
                try {
                   out = new FileOutputStream(file);
                } catch (FileNotFoundException e) {
                   e.printStackTrace();
```

```
                }
                ByteArrayOutputStream bitmapStream =
ZSingleton.getInstance().getMyByteArrayOutputStream();
                try {
                  if (out != null) {
                    out.write(bitmapStream.toByteArray());
                    out.close();
                  }
                } catch (IOException e) {
                  e.printStackTrace();
                }
                if (ZSingleton.getInstance().getCount() > 0) {
                  Integer pos;
                  pos = ZSingleton.getInstance().getCount() - 1 -
ZSingleton.getInstance().getEdit();
                  if (pos >= 0) {
                    ed.putString("Backgroundurl" +
ZSingleton.getInstance().getBckPos()[pos]
                        , backPic.getAbsolutePath());
                    ed.commit();
                  }
                }
              }
              //sends intent to next activity
             if (ZSingleton.getInstance().getEdit() < skaits - 1) {
               ZSingleton.getInstance().setEdit(ZSingleton.getInstance().getEdit() + 1);
                Intent intent = new Intent(getApplicationContext(),ZGetFiles.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                intent.putExtra("isBackground", true);
                ZSingleton.getInstance().setCheck(false);
                startActivity(intent);
                finish();
              } else {
                ZSingleton.getInstance().setCheck(false);
                ZSingleton.getInstance().setEdit(0);
                ZSingleton.getInstance().setListsToNull();
                ZSingleton.getInstance().setPrefsId(null);

                if (ZSingleton.getInstance().getFromTitle()) {
                  ZSingleton.getInstance().setFromTitle(false);
                  ZSingleton.getInstance().setRestart(false);
                  finish();
                } else {
                  ZSingleton.getInstance().setRestart(true);
                  ZSingleton.getInstance().setFromTitle(false);
```

```
                    ZSingleton.getInstance().setWidgets(true);
                    Intent intent = new Intent(getApplicationContext(), ZList.class);
                    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    intent.putExtra("isBackground", false);
                    startActivity(intent);
                    finish();
                }
            }
        }
    }
  }
}
```

Kvalifikācijas darbs „**Informatīvā paneļa "InfoDesk" tīkla funkcionaitātes 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: ***Toms Kirtovskis*** **_____ ___.05.2015**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs**: Dr. Inž Artis Teilāns _____ ___.05.2015.**

Recenzents: ***Dr. Dat Zane Bičevska***

Darbs iesniegts 01.06.2015.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): _____