LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

“Near Field Clues”, uz Ģeolokāciju un NFC tehnoloģijām balstīta izpētes spēle

KVALIFIKĀCIJAS DARBS

Autors: **Hermanis Verhoustinskis**

Studenta apliecības Nr.: hv13001

Darba vadītājs: Dr.dat. Guntis Arnicāns

RĪGA 2016

Anotācija

Hermaņa Verhoustinska kvalifikācijas darbā – “Near Field Clues”, uz Ģeolokāciju un NFC tehnoloģijām balstīta izpētes spēle – ir izstrādāta divplatformu lietotne Windows un Android lietotājiem. Lietotnes mērķis – nodrošināt izklaidējošu un virtuāli atalgojošu spēli, kas tās lietotājus mudinātu izpētīt un apmeklēt dažādus punktus pasaulē. Lietotne balstās uz divām galvenajām tehnoloģijām – Ģeolokāciju (tehnoloģija, kas izmantojot GPS un citus atrašanās vietas risinājumus, var noteikt ierīces atrašanās vietu uz Zemes divdimensiju plaknē) un NFC(ļoti nelielu attālumu bezvadu datu pārraides tehnoloģija).

Izmantojot šis tehnoloģijas kvalifikācijas darbā izstrādātais risinājums ļauj lietotājiem ierakstīt, lasīt un kartē atainot NFC birkas. Šīs birkas kalpo kā identifikatori, ko lietotne interpretē un veic pieprasītās darbības.

Lietotne izstrādāta pēc spirāles izstrādes modeļa un kvalifikācijas darbs sedz tikai pirmo izstrādes iterāciju, bet tajā tiks minētas ar nākotnes lietotnes perspektīvas.

Lietotne izstrādāta izmantojot Qt bibliotēkas un C++ priekš aizmugursistēmas un QML priekš saskarnes.

Atslēgas vārdi: C++, Qt, QML, NFC, Ģeolokācija, cross-platform

Abstract

"Near Field Clues" is a cross-platform application for Windows Desktop and android devices, structured around NFC (Near Flied Communication an extremely short-range wireless technology) and Geolocation (identification of real-world geographic location of an object) technologies. This application aims to offer an entertaining game-like way of discovering and exploring the world around you.

The conjunction of the two technologies in this solution allows writing and reading NFC tags which are the used as global geolocation object. The tag serves as an identification to be used internally in the application and perform requested procedures.

The application has been developed using spiral methodology, and mainly this document only covers the first iteration of the application, but there will be mentions of future perspectives of the product.

The solution has been developed using Qt libraries und C++ for the back-end of the application, and QML has been used to build the user interface.

Keywords: C++, Qt, QML, NFC, Geolocation, cross-platform

Saturs

Saturs 4

1. Ievads 8

1.1. Programmatūras konceptuālais apraksts 8

1.2. Konceptuālie standartscenāriji 8

1.3. Virspusējs arhitektūras apraksts 10

2. Programmatūras prasību specifikācija 12

2.1. Nolūks 12

2.2. Darbības sfēra 12

2.3. Definīcijas, akronīmi un saīsinājumi 12

2.4. Vispārējais apraksts 13

2.4.1. Produkta perspektīva 13

2.4.2. Produkta funkcijas 13

2.4.3. Lietotāju raksturzīmēs 13

2.4.4. Vispārējie ierobežojumi 13

3. Funkcionālās prasības 14

3.1. Birku funkcijas 14

3.1.1. Nolasīt birku 14

3.1.2. Ierakstīt birku 14

3.2. Piedzīvojumu funkcijas 15

3.2.1. Jauna piedzīvojuma izveide 15

3.2.2. Piedzīvojuma inicializēšana 16

3.2.3. Piedzīvojuma pievienošana lietotāja piedzīvojumu sarakstam 17

3.2.4. Piedzīvojuma izpildīšana 18

3.2.5. Piedzīvojuma dzēšana 19
3.2.6. Piedzīvojumu uzskaite 19
3.2.7. Piedzīvojuma apskatīšana 20
3.3. Lietotāju funkcijas 21
3.3.1. Lietotāja izveidošana 21
3.3.2. Lietotāja pieslēgšanās 22
3.3.3. Lietotāja profila apskate 22
3.3.4. Uzcelt lietotāja izveidotos piedzīvojumus 22
3.3.5. Uzcelt lietotāja izpildītos piedzīvojumu 24
3.3.6. Uzcelt līderu tabulu 24
3.4. Administratora funkcijas 25
3.4.1. Lietotāja dzēšana 25
3.4.2. Piedzīvojumu dzēšana 25
3.5. Datu konceptuālais modelis 26
3.6. Lietotāja saskarne 26
4. Nefunkcionālās prasības 27
4.1. Veiktspējas prasības 27
4.2. Drošība 27
5. Programmatūras projektējuma apraksts 28
5.1. Nolūks 28
5.2. Programmatūras datu plūsmas diagrammas 28
5.3. Funkcionalitātes piemēri 32
5.3.1. Izveidot jaunu piedzīvojumu un to inicializēt 32
5.3.2. Izpildīt piedzīvojumu 33
5.4. Lietotnes funkcionālais projektējums 33

5.4.1. Klase UserHandler 33
5.4.2. Klase AdventureHandler 37
5.4.3. Klase AdventureOnUserData 43
5.4.4. Klase LeaderboardData 44
5.4.5. Klase MapItemData 46
5.4.6. Klase System 47
5.4.7. Klase NfcDb 48
5.4.8. Klase NfcHandler 48
5.5. Skatu izkārtojumi 52
5.5.1. Saskarņu diagrammas 52
5.5.2. Main.qml 54
5.5.3. MainMenu.qml 54
5.5.4. MapPopup.qml 56
5.5.1. HandleUser(mainUserHandle) 56
5.5.2. HandleAdventure (thisAdvendture) 57
5.5.3. System (thisSystem) 57
5.5.4. CreateAdventures.qml 57
5.5.5. Login.qml 58
5.5.6. Register.qml 58
5.5.7. User.qml 58
5.5.8. Leaderboard.qml 59
5.5.9. SeeAdventure.qml 59
6. Projekta organizācija 61
6.1. Konfigurāciju pārvaldība 61
6.2. Kvalitātes nodrošināšanā 62

7. Darbietilpības novērtējums 62
8. Testēšanas dokumentācija 63
8.1. Testēšanas scenāriji 63
8.1.1. Jebkurš var izveidot jaunu lietotāju 63
8.1.2. Lietotājs var pieslēgties ar esošu lietotāju 64
8.1.3. Lietotājs apskata savu profilu 64
8.1.4. Lietotājs apskata savus piedzīvojumus 65
8.1.5. Lietotājs apskata Līderu tabulu 65
8.1.6. Lietotājs izveido jaunu piedzīvojumu 65
8.1.7. Lietotājs apskata konkrētu piedzīvojumu 66
8.1.8. Lietotājs izpilda piedzīvojumu 66
8.1.9. Lietotājs inicializē piedzīvojumu 67
8.1.10. Lietotājs dzēš savu piedzīvojumu 67
9. Nobeigums 68
Izmantotā literatūra 69
Pielikums 70
NFClues lietotnes ceļvedis 70
Programmprodukta pirmkoda fragments 71

1. Ievads

Šis nodaļas nolūks ir iepazīstināt lasītāju ar virspusējajām/konceptuālajām programmatūras darbībām un scenārijiem un sniegt ieskatu risinājumā arhitektūrā.

1.1. Programmatūras konceptuālais apraksts

Lietotne sastāv no diviem galvenajiem modeļiem – piedzīvojums un lietotājs.

Piedzīvojums ir objektu kopa, kas lietotājam tiek vizualizēta kartē kā ģeolokācijas objekts. Tas sevī ietver:

- Lat Long koordinātas – ģeolokācijas koordināšu garuma un platuma grādu komponentes,
- norādi uz NFC birku – identifikators pēc kura ir iespējams identificēt piedzīvojumu,
- papildus raksturdatus – piedzīvojuma nosaukums, apraksts un citi, aprakstīti tālāk dokumentā.

Programmatūra gala lietotājam tiek attēlota, kā atzīme kartē, kuru atverot tiek attēloti raksturdati. Piedzīvojumam var būt piesaistīta NFC birka (Neliels NFC raidītājs vai uztvērējs, kurā var ievietot un lasīt nelielu daudzumu informācijas, pār nelieliem attālumiem).

Piedzīvojumam ir divas stadijas - neinicializēts (piedzīvojumam ir specificēti raksturdati, bet tam vēl nav piesaistītas ģeolokācijas koordinātas un NFC birkas identifikators, līdz tas tiek inicializēts tas ir pieejams tikai piedzīvojuma izveidotājam) un inicializēts (piedzīvojumam ir piešķirtas ģeolokācijas koordinātas un tas ir sasaistīts ar unikālu NFC birkas identifikatoru, tas kļūst publiski pieejams visiem lietotnes lietotājiem kā ģeolokācijas objekts).

Lietotājs ir lietotnes gala lietotājs ar iespējām veidot jaunus piedzīvojumus un izpildīt esošos.

1.2. Konceptuālie standartscenāriji

Šeit tiek aprakstīti tikai sarežģītākie scenāriji, lai lasītājam sniegtu darbplūsmas piemēru skaidrākam lietotnes darbības kontekstam. Lietotāja modeļa scenāriji, detalizēta realizācija, kā arī pārējo funkciju detalizēts apraksts pieejams tālākās nodaļās.

Konceptuāls scenārijs jauna inicializētas piedzīvojuma izveidošanai ir sekojošs:

1. Lietotājs pieslēdzas lietotnei

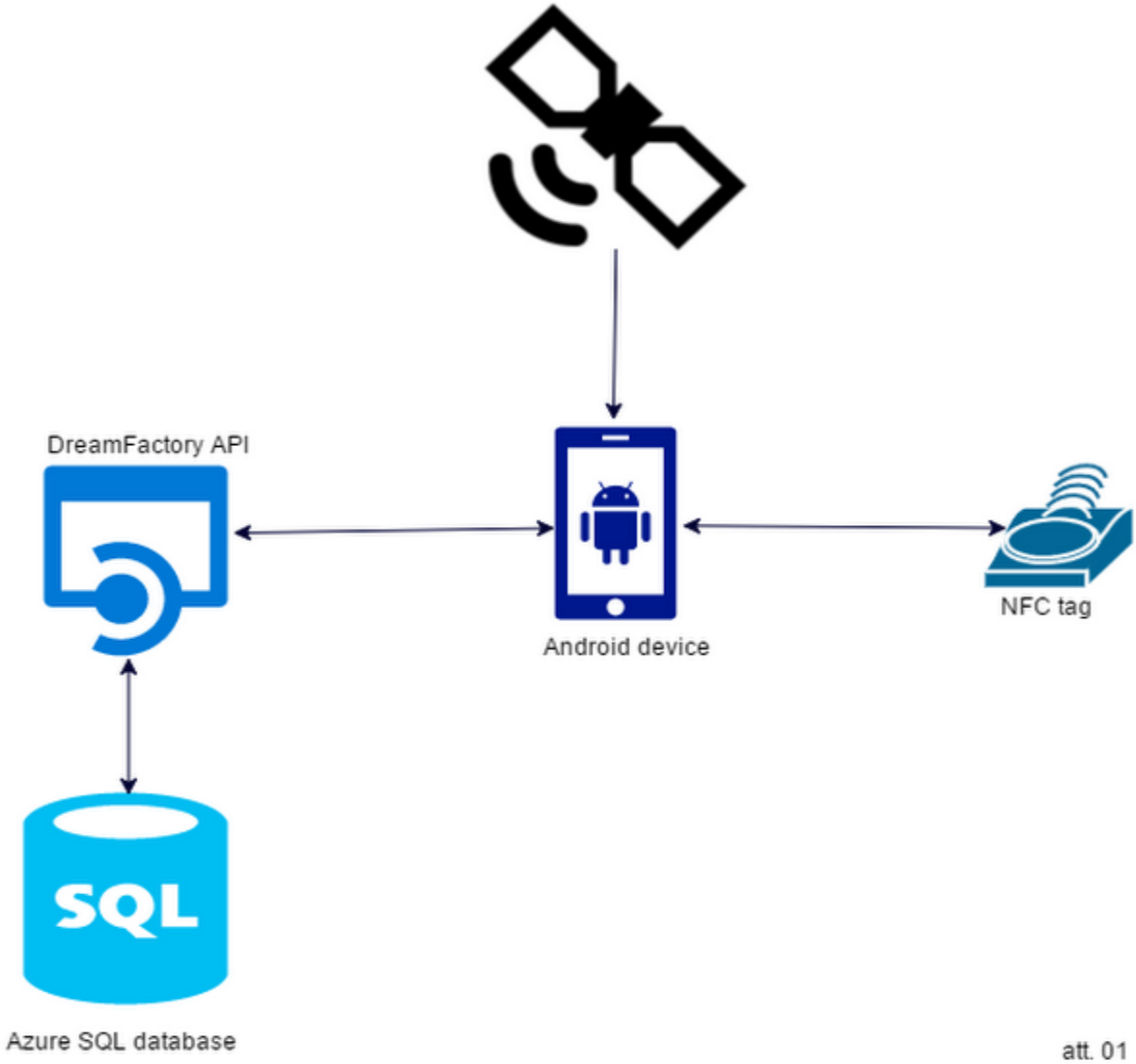
2. Lietotājs izveido jaunu piedzīvojumu
3. Lietotājs savu piedzīvojumu sarakstā izvēlas jauno piedzīvojumu un izpilda inicializēt darbību
4. Lietotājs pēc pieprasīju ierīcei norāda birku
5. Ierīce nolasa pašreizējās Lat Long koordinātas
6. Ierīce ieraksta birkā identifikatoru un piedzīvojumam piešķir Lat Long koordinātas

Konceptuāls scenārijs piedzīvojuma izpildīšanai:

1. Lietotājs atrod piedzīvojuma identificējošo birku
2. Lietotājs pieslēdzas lietotnei
3. Lietotājs novieto ierīci tuvu NFC birkai
4. Ierīce saņem ziņu no NFC birkas, tā tiek identificēta un piesaistītais piedzīvojums tiek izpildīts.

1.3. Virspusējs arhitektūras apraksts

Virspusējs programmatūras arhitektūras attēlojums redzams att. 01



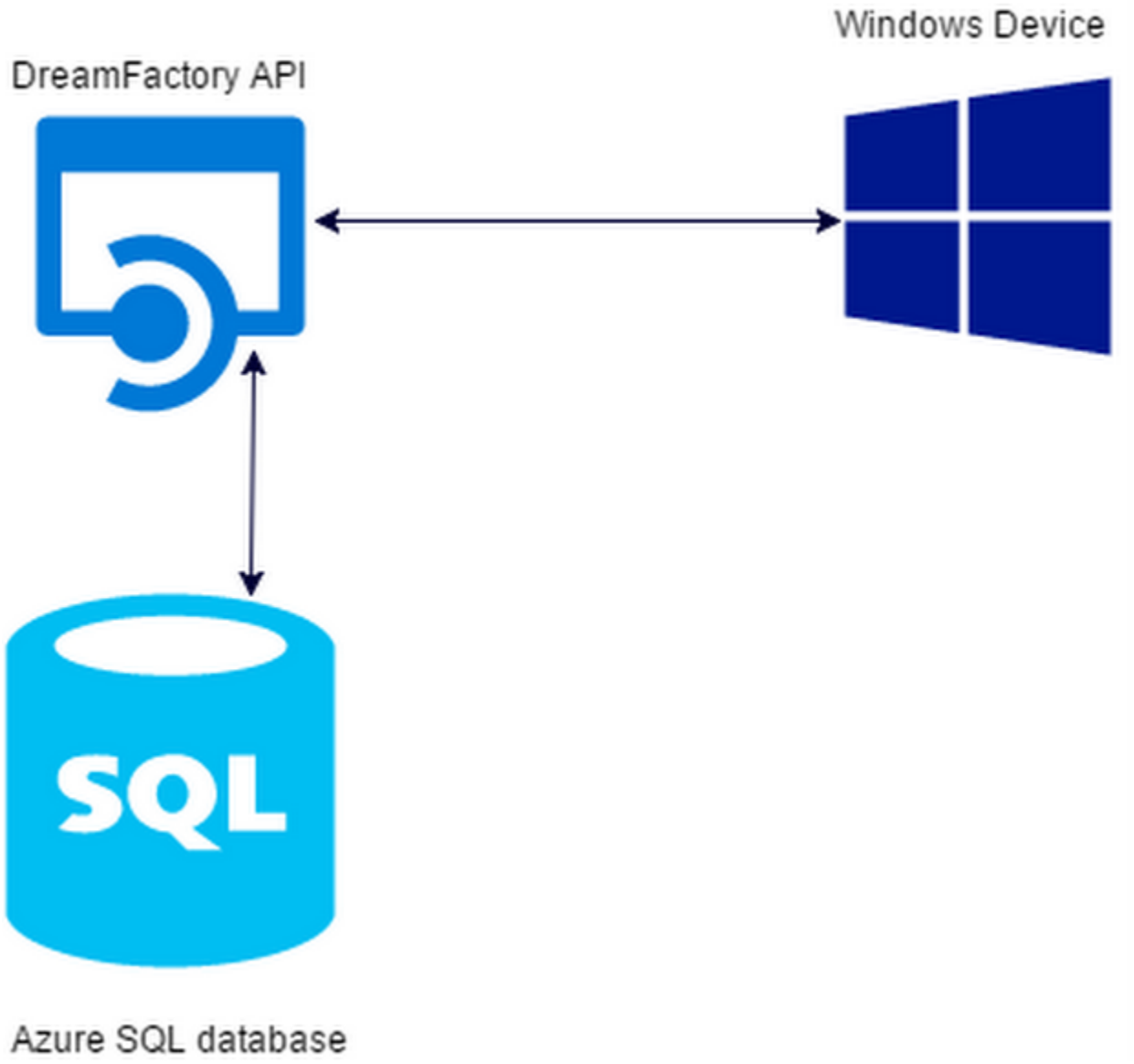
att. 01

Lai programmatūra pareizi funkcionētu visiem šiem resursiem jābūt pieejamiem.

Visām darbībām ar datubāzi ir jānotiek caur DreamFactory API.

Lai uztvertu NFC birku ierīcei jābūt pieejamam NFC savienojumam un tam jābūt ieslēgtam, lietotnei nav iespējas to automātiski ieslēgt.

Windows platformas arhitektūras izskatās ļoti līdzīgi, bet tajā nav pieejama pašreizējā atrašanās vieta, ja tam nav pieejams GPS serviss, kā arī nav pieejama darbība ar NFC birkām. Galvenā loma Windows videi ir atvieglota administrēšana.



2. Programmatūras prasību specifikācija

2.1. Nolūks

Šī nodaļa rakstīta balstoties uz programmatūras prasību specifikācijas ceļvedi(LVS 68:1996). Tā raksturo funkcionālas prasības izstrādājamajās programmas pirmajā ciklā, tās saistības ar ārējajiem resursiem un virspusēji apraksta dizaina prasības.

2.2. Darbības sfēra

Programmatūra paredzēta Androīd lietotājiem ar mobilajām ierīcēm, kurām ir pieejams NFC, kā arī ir pieejama Windows lietotājiem. Tās mērķis ir izklaide un izpēte, sniedzot lietotājiem virtuālu atalgojumu par izpildītiem piedzīvojumiem un dodot lietotājiem iespēju veidot pašiem savus piedzīvojumus.

2.3. Definīcijas, akronīmi un saīsinājumi

NFC – tehnoloģija, standartu kopa, kas nodrošina maza attāluma bezvadu datu pārraidi (angļu Near Field Communication)

DB – datu bāze

DreamFactory API – WEB serviss, virtuālā mašīna, kas automātiski piesaistās datubāzes galapunktiem un padara tos pieejamus caur WEB pieprasījumiem

Qt – C++ bibliotēka, ietvars, kas ļauj ērti veidot divplatformu lietotnes

Git – Versiju kontroles rīks

ID – Identifikators

Lat – Ģeolokācijas koordinātes platuma grādu komponente

Long – Ģeolokācijas koordinātes garuma grādu komponente

2.4. Vispārējais apraksts

2.4.1. Produkta perspektīva

Caur Google Play Store publiski pieejama bezmaksas izklaides lietotne, kas tās uzturēšanas izmaksas nodrošinātu ar specializētu NFC birku pārdošanu.

2.4.2. Produkta funkcijas

Pirmā programmas iterācija nodrošinās funkcionalitāti:

- Piedzīvojumu izveidošana, inicializēšana, attēlošana kartē, izpildīšana
- Punktu atalgošana par izpildītiem piedzīvojumiem
- Piedzīvojumu vēstures apskatīšana
- Paša izveidoto piedzīvojumu apskatīšana
- Jaunu lietotāju veidošana un pārvaldīšana
- Esošu lietotāju pieslēgšanās
- Lietotāju un piedzīvojumu administrēšana

2.4.3. Lietotāju raksturzīmēs

Pilnai programmatūras funkcionalitātei lietotājam ir nepieciešams mobilais tālrunis ar Androīd operētājsistēmu, tam jābūt NFC pieejamībai, kā arī šim tālrunim darbības laikā jābūt interneta pieejai.

2.4.4. Vispārējie ierobežojumi

- Programmatūrai jābūt uzinstalētai uz ierīces
- Ierīcei, uz kuras tiek lietota programmatūra, jābūt ar NFC pieejamību
- Ierīcei, uz kuras tiek lietota programmatūra, jābūt interneta pieejai
- Labākai ģeolokācijas precizitātei ierīcei, uz kuras tiek lietota programmatūra, jābūt pieejamam GPS

3. Funkcionālās prasības

3.1. Birku funkcijas

3.1.1. Nolasīt birku

Funkcija nolasa informāciju birkā.

Ievade	Parametri:
Apstrāde	Validē:
Izvade	Paziņojums, ka birka nolasīta veiksmīgi.

3.1.2. Ierakstīt birku

Funkcija ieraksta informāciju birkā.

Ievade	Parametri: • Birkas ID
Apstrāde	Validē:
Izvade	Paziņojums, ka birka ierakstīta veiksmīgi.

3.2. Piedzīvojumu funkcijas

3.2.1. Jauna piedzīvojuma izveide

Ievade	Parametri: • Piedzīvojuma nosaukums (ASCII simboli garumā līdz 256 simboliem) • Piedzīvojuma apraksts (ASCII simboli garumā līdz 4084 simboliem) • Atalgojums (Skaitlis ,Atalgojums par piedzīvojuma izpildīšanu) • Pavediens (ASCII simboli garumā līdz 4084 simboliem , papildinformācija, ko lietotājs var izmantot, ja rodas sarežģījumi atrast birku)
Apstrāde	Validē: • Vai aizpildīts piedzīvojumu nosaukums un atalgojums • Vai padots lietotāja ID • Vai šim lietotāju nav piedzīvojums ar tādu pašu nosaukumu Izveidots neinicializēts piedzīvojums, redzams tikai lietotājam, kurš to izveidoja skatā “mani piedzīvojumi”.
Izvade	Ja pārkāpta kāda validācija tiek parādīt kļūdas paziņojums. Ja dati ir pareizi, lietotājam tiek paziņots par to, ka piedzīvojums izveidots un viņš tiek novadīts uz “mani piedzīvojumi” lapu. Lietotājam jaunizveidotais piedzīvojums parādās sarakstā “mani piedzīvojumi”, kā neinicializēts.

3.2.2. Piedzīvojuma inicializēšana

Funkcija ļauj izveidotos piedzīvojumus inicializēt ar telefona novietošanu tieši uz birkas, kā rezultātā lietotājs tiek izvadīts cauri inicializēšanas procesam.

Ievade	Parametri: • Birkas ID • Piedzīvojuma ID
Apstrāde	Validē: • Vai padots birkas ID • Vai padots piedzīvojuma ID • Vai padotas koordinātas Vispirms nolasa birku (funkcija 3.1.1), ja tajā nav informācija, tad tiek ierakstīts padotais birkas ID (funkcija 3.1.2), kad tas ir izdarīts, tiek nolasītas pašreizējās ierīces koordinātas un piedzīvojuma ieraksts tiek atjaunināts ar šo birkas ID un patreizējajām Lat Long koordinātām.
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ir pareizi, lietotājam tiek paziņots par to, ka piedzīvojums inicializēts un tagad tas ir publiski pieejams. Lietotājs tiek novadīts uz sākumlapu.
Komentārs	Šī funkcija nav pieejama, ja nav ieslēgts NFC savienojums.

3.2.3. Piedzīvojuma pievienošana lietotāja piedzīvojumu sarakstam

Funkcija pievieno piedzīvojumu jau esošajam sarakstam, lai tas nav jāpārlādē.

Ievade	Parametri: • ID • Vārds • Apraksts • Status • Pavediens
Apstrāde	Šie dati tiek pievienoti jau gatavajai tabulai lietotāja piedzīvojumi.
Izvade	Lietotājs tiek novadīts uz lietotāja piedzīvojumu sarakstu.
Komentārs	

3.2.4. Piedzīvojuma izpildīšana

Funkcija ļauj izveidotos, inicializētos piedzīvojumus izpildīt ar vienkāršu tālruņa pārvilkšanu pāri NFC birkai un par tiem saņemt atalgojumu.

Ievade	Parametri: • Birkas ID • Lietotāja ID
Apstrāde	• Validē: • Vai padots Birkas ID • Vai padots Lietotāja ID • Vai padotais Birkas ID atbilst kādam piedzīvojumam Ja šim birkas ID ir atrasts attiecīgs piedzīvojums, no tā tiek nolasīta atalgojuma informācija, tā tiek pieskaitīta lietotājam. Šis piedzīvojums tiek reģistrēts kā izpildīts šim lietotājam.
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ir pareizi, lietotājam tiek paziņots par to, ka piedzīvojums izpildīts un tagad tas tiek pievienots izpildīto piedzīvojumu sarakstam.
Komentārs	Šī funkcija nav pieejama, ja nav ieslēgts NFC savienojums.

3.2.5. Piedzīvojuma dzēšana

Funkcija ļauj lietotājiem dzēst savus un administratoriem dzēst visus piedzīvojumus.

Ievade	Parametri: • Piedzīvojuma ID
Apstrāde	Validē: • Vai padots piedzīvojuma ID • Vai lietotājs ir piedzīvojuma īpašnieks vai administrators
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ir pareizi, lietotājam tiek paziņots par to, ka piedzīvojums izdzēsts un tas pazūd no piedzīvojumu saraksta.
Komentārs	

3.2.6. Piedzīvojumu uzskaite

Funkcija ļauj lietotājiem apskatīt inicializētus piedzīvojumus attēlotus kartē.

Ievade	
Apstrāde	Tiek pieprasīts saraksts ar visiem inicializētiem piedzīvojumiem un to Lat Long koordinātām, tie tiek izkārtoti pa karti.
Izvade	Tiek atainota Google maps karte ar centrējumu uz lietotāja pašreizējo atrašanās vietu, kustība kartē ir brīva, tajā ir iespējams pietuvināt un attālināt. Kartē tiek attēloti inicializētie piedzīvojumi.
Komentārs	

3.2.7. Piedzīvojuma apskatīšana

Funkcija ļauj lietotājiem apskatīt inicializētus piedzīvojumus. Redzēt atalgojumu un pavedienu.

Ievade	Parametri: Piedzīvojuma ID
Apstrāde	Validē: Vai padots piedzīvojuma ID
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ir pareizi, lietotājam tiek paradīti piedzīvojuma rakstur dati un iespēja aplūkot pavedienu. Ja piedzīvojums nav inicializēts tad lietotājam ir iespēja to inicializēt šajā skatā. Ja lietotājs, kas skatās piedzīvojumu ir administrators vai piedzīvojuma īpašnieks, tad ir iespēja dzēst šo piedzīvojumu.
Komentārs	

3.3. Lietotāju funkcijas

3.3.1. Lietotāja izveidošana

Ievade	Parametri: • Lietotājvārds (ASCII simboli garumā līdz 30 simboliem) • E-pasts • Parole (ASCII simboli garumā līdz 30 simboliem) • Parole atkārtoti (ASCII simboli garumā līdz 30 simboliem, garāka par 6 simboliem)
Apstrāde	Validē: • Vai lietotājvārds ievadīts un ir unikāls • Vai e-pasts ievadīts un ir unikāls • Vai parole ievadīta un sakrīt ar paroli atkārtoti un ir garāka par 6 simboliem
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja Dati ievadīti pareizi, tad lietotājs tiek izveidots un notiek automātiska pieslēgšanas.

3.3.2. Lietotāja pieslēgšanās

Funkcija pieslēdzas esošam lietotājam.

Ievade	Parametri: • Lietotājvārds (ASCII simboli garumā līdz 30 simboliem) • Parole (ASCII simboli garumā līdz 30 simboliem)
Apstrāde	Validē: • Vai lietotājvārds ievadīts • Vai parole ievadīta • Vai SHA:1 vērtība šai parolei sakrīt ar vērtību atgrieztu no DreamFactory
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja Dati ievadīti pareizi, tad lietotājs pieslēdzas, izsauc funkcijas 3.3.4, 3.3.5, 3.3.6 un tiek novadīts uz sākumlapu.

3.3.3. Lietotāja profila apskate

Funkcija ļauj apskatīt esoša lietotāja profilu, tā izpildītos piedzīvojumus un sakrāto punktu skaitu, kā arī pašreizējo vietu.

Ievade	Parametri: • Lietotāja ID
Apstrāde	Validē: • Vai padots lietotāja ID
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ievadīti pareizi, tad lietotājs tiek aizvests uz izvēlētā lietotāja profilu.

3.3.4. Uzcelt lietotāja izveidotos piedzīvojumus

Funkcija ļauj apskatīt esoša lietotāja izveidoto piedzīvojumu sarakstu, kādu no tiem izvelēties, lai apskatītos specifiku piedzīvojumu.

Ievade	Parametri: • Lietotāja ID
Apstrāde	Validē: • Vai padots lietotāja ID Ja lietotājs izvēlās kādu no šiem piedzīvojumiem tas tiek padots uz "apskatīt piedzīvojumu" (funkcija 3.2.6)
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ievadīti pareizi, tad tiek izveidots saraksts ar lietotāja izveidotajiem piedzīvojumiem
Komentārs	

3.3.5. Uzcelt lietotāja izpildītos piedzīvojumu

Funkcija ļauj apskatīt esoša lietotāja izpildītos piedzīvojumus.

Ievade	Parametri: • Lietotāja ID
Apstrāde	Validē: • Vai padots lietotāja ID Ja lietotājs izvēlās kādu no šiem piedzīvojumiem tas tiek padots uz apskatīt piedzīvojumu (funkcija 3.2.6)
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ievadīti pareizi, tad tiek izveidots saraksts ar lietotāja izpildītajiem piedzīvojumiem.
Komentārs	

3.3.6. Uzcelt līderu tabulu

Funkcija ļauj apskatīt pašreiz vadošos lietotājus pēc punktu skaita.

Ievade	
Apstrāde	
Izvade	Izveido līderu tabulu, kas pieejama līderu skatā.
Komentārs	

3.4. Administratora funkcijas

3.4.1. Lietotāja dzēšana

Funkcija ļauj administratoriem dzēst lietotājus.

Ievade	Parametri: • Lietotāja ID
Apstrāde	Validē: • Vai padots lietotāja ID • Vai funkcijas pieprasītājs ir administrators
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ievadīti pareizi, tad lietotājs tiek dzēsts
Komentārs	

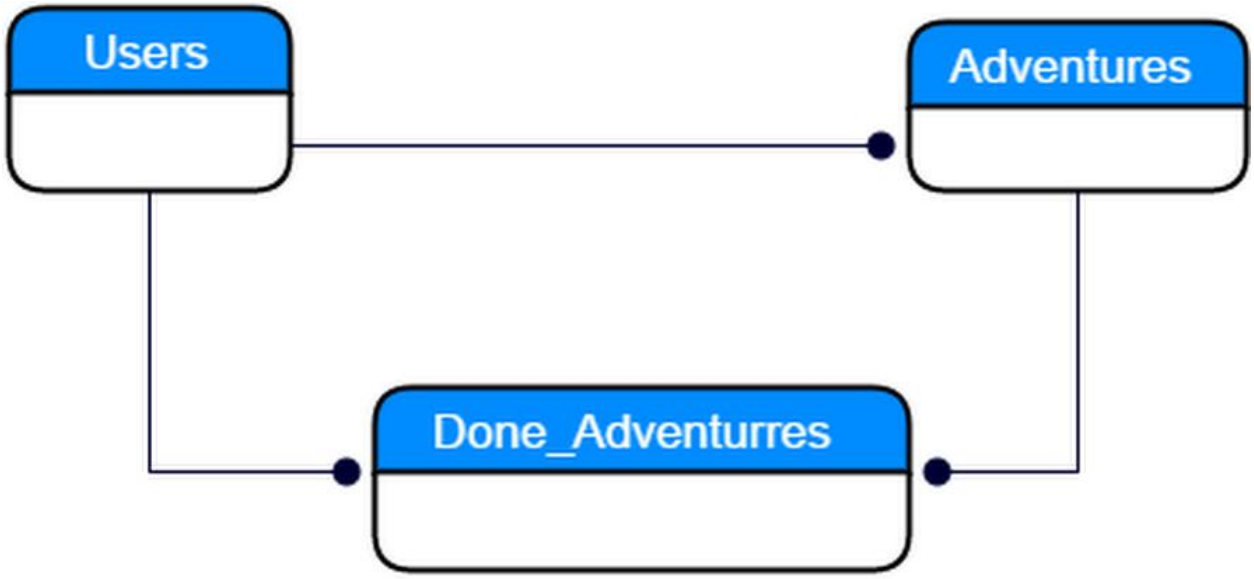
3.4.2. Piedzīvojumu dzēšana

Funkcija ļauj administratoriem dzēst piedzīvojumus.

Ievade	Parametri: • Piedzīvojuma ID
Apstrāde	Validē: • Vai padots piedzīvojuma ID • Vai funkcijas pieprasītājs ir administrators
Izvade	Ja pārkāpta kāda validācija tiek parādīts kļūdas paziņojums. Ja dati ievadīti pareizi, lietotājam tiek paziņots par to, ka piedzīvojums izdzēsts un tas pazūd no piedzīvojumu saraksta.
Komentārs	

3.5. Datu konceptuālais modelis

Pirmajā izstrādes ciklā konceptuālais modelis ir ļoti vienkāršs, nodrošinot tikai pamat funkcionalitāti.



att. 07

3.6. Lietotāja saskarne

Ērtākā un pilnākā saskarne tiek veidota un pielāgota Androīd vidēm, bet tā ir arī pieejama Windows ierīcēm, lai gan tā būs jūtami "nedraudzīgāka", lai tā pareizi funkcionētu, jābūt pieejai pie interneta, lai varētu veidot karti, tajā izvietot piedzīvojumu, kā arī savienojumam ar DreamFactory API, lai atainotu aktuālos datus no datubāzes.

Pirmajā iterācijā saskarnei jāizmanto ierīces noklusētās vērtības atainojot visus saskarnes elementus.

4. Nefunkcionālās prasības

4.1. Veiktspējas prasības

Pirmajā iterācijā sistēmai jāspēj nodrošināt netraucētu pilna laika savienojumu ar datubāzi vismaz 20 lietotājiem vienlaicīgi.

4.2. Drošība

Lai lietotni izmantotu lietotājam ir jāautentificējas ievadot paroli, kas ir attiecīga šim lietotājvārdam. Parole tiek glabāta datubāzē šifrētā formātā.

Pieeja datubāzei notiek tikai caur DreamFactory API.

5. Programmatūras projektējuma apraksts

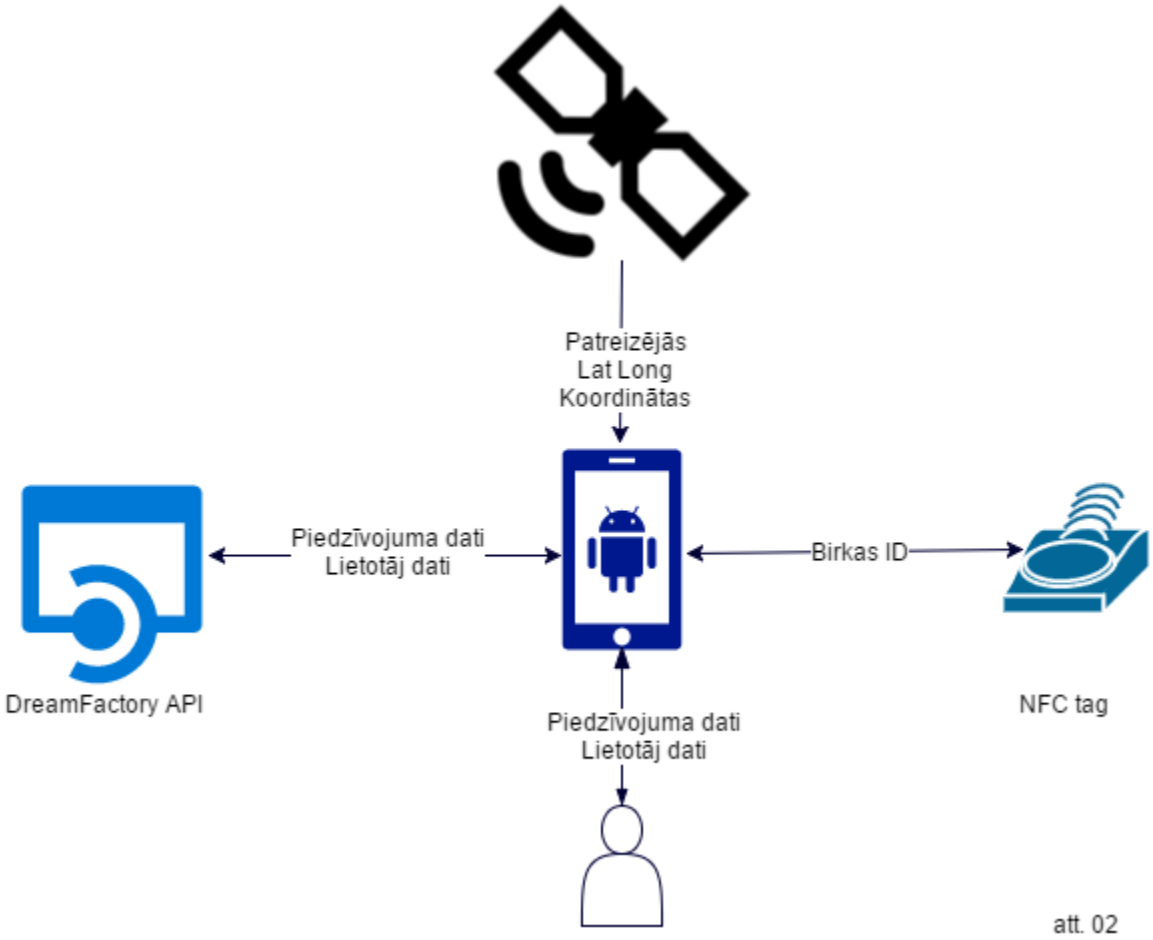
5.1. Nolūks

Programmatūras projektējuma apraksta nolūks ir sniegt ieskatu produkta prasību specifikācijā, definētās prasības ir realizētas programmprodukta izstrādē.

5.2. Programmatūras datu plūsmas diagrammas

Katra funkcija norādīt datu plūsmas diagramma sevī ietver kļūdas paziņojuma pēc noklusējuma, tas datu plūsmās netiek ietverts.

0. līmeņa datu plūsmas diagramma redzama att. 02

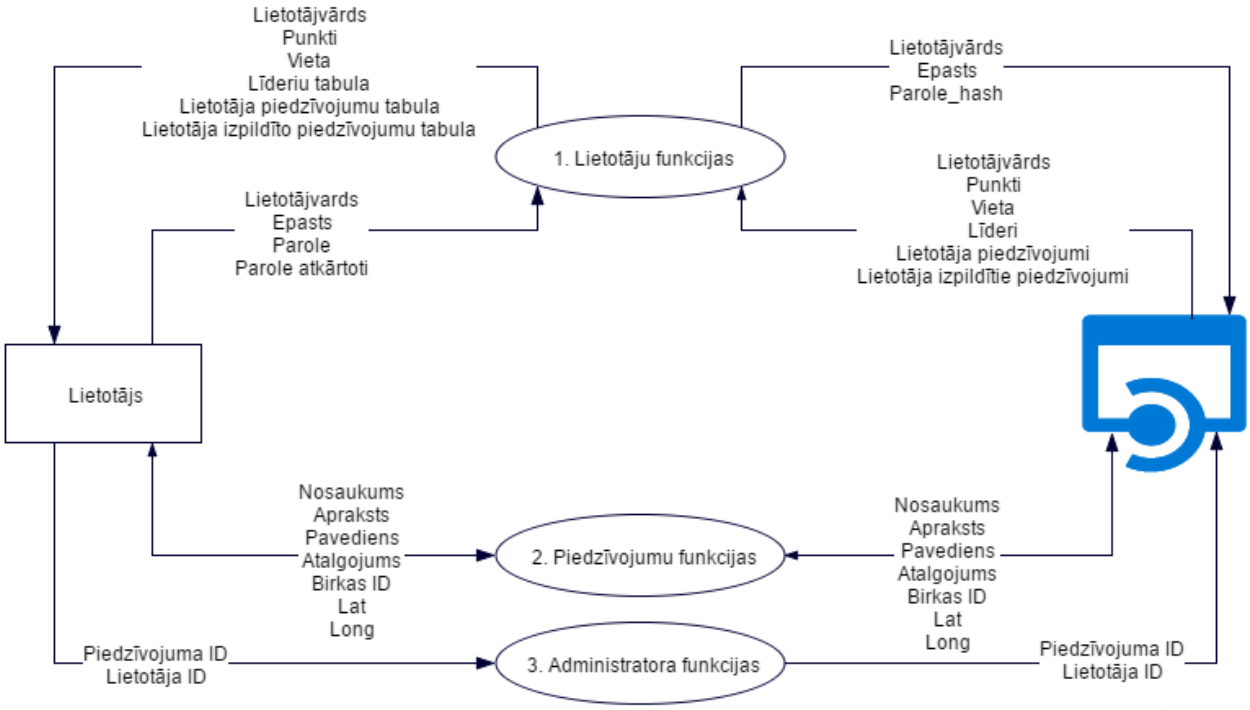


att. 02

(0. līmenī dati aprakstīt virspusēji, tie tiks precizēti tālākās diagrammās)

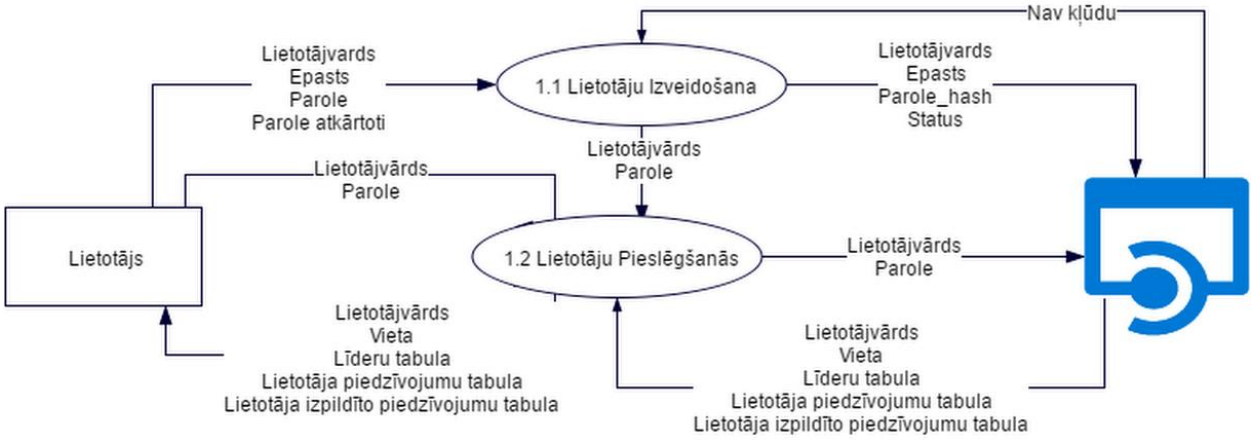
(1.

1. līmeņa datu plūsmas diagramma redzama att. 03



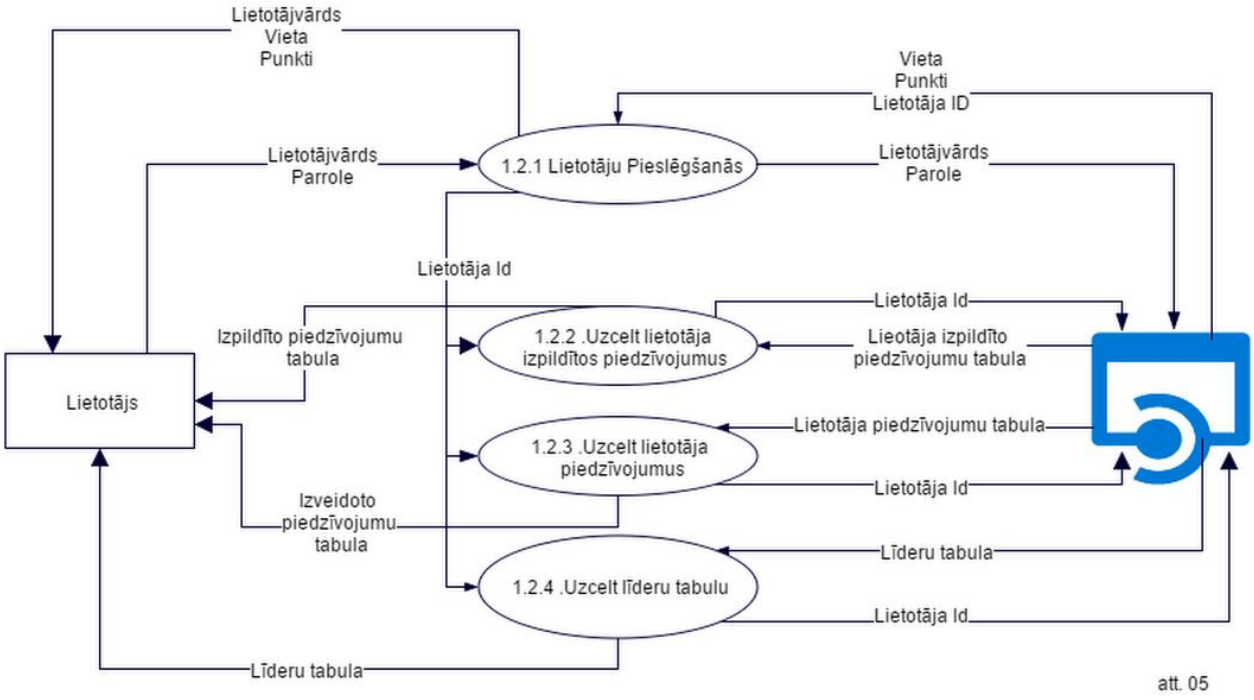
att. 03

2 līmeņa lietotāju funkcijas datu plūsmas diagramma redzama att. 04

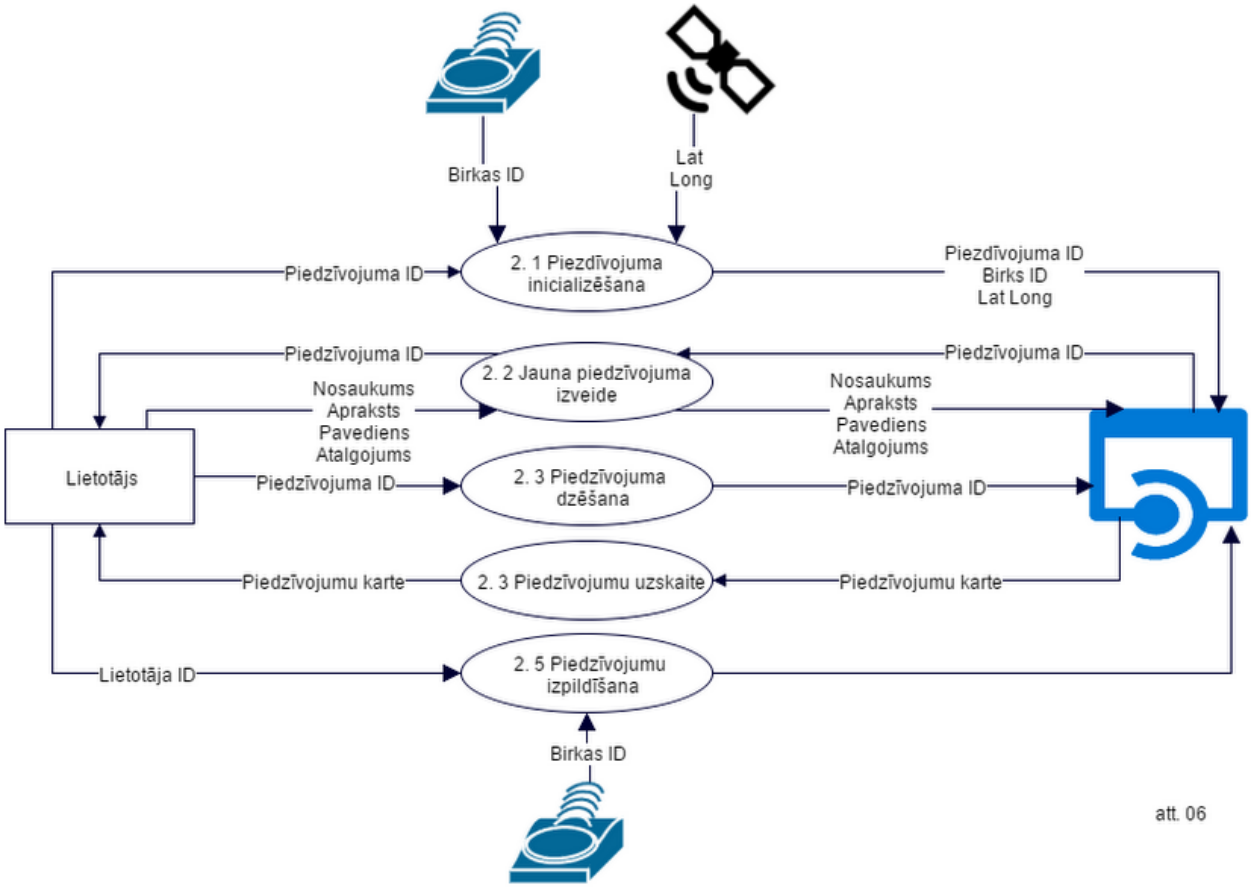


att. 04

3. līmeņa lietotāju funkcijas “Lietotāja pieslēgšanās” datu plūsmas diagramma redzama att. 04



2. līmeņa piedzīvojumu funkcijas datu plūsmas diagramma redzama att. 06

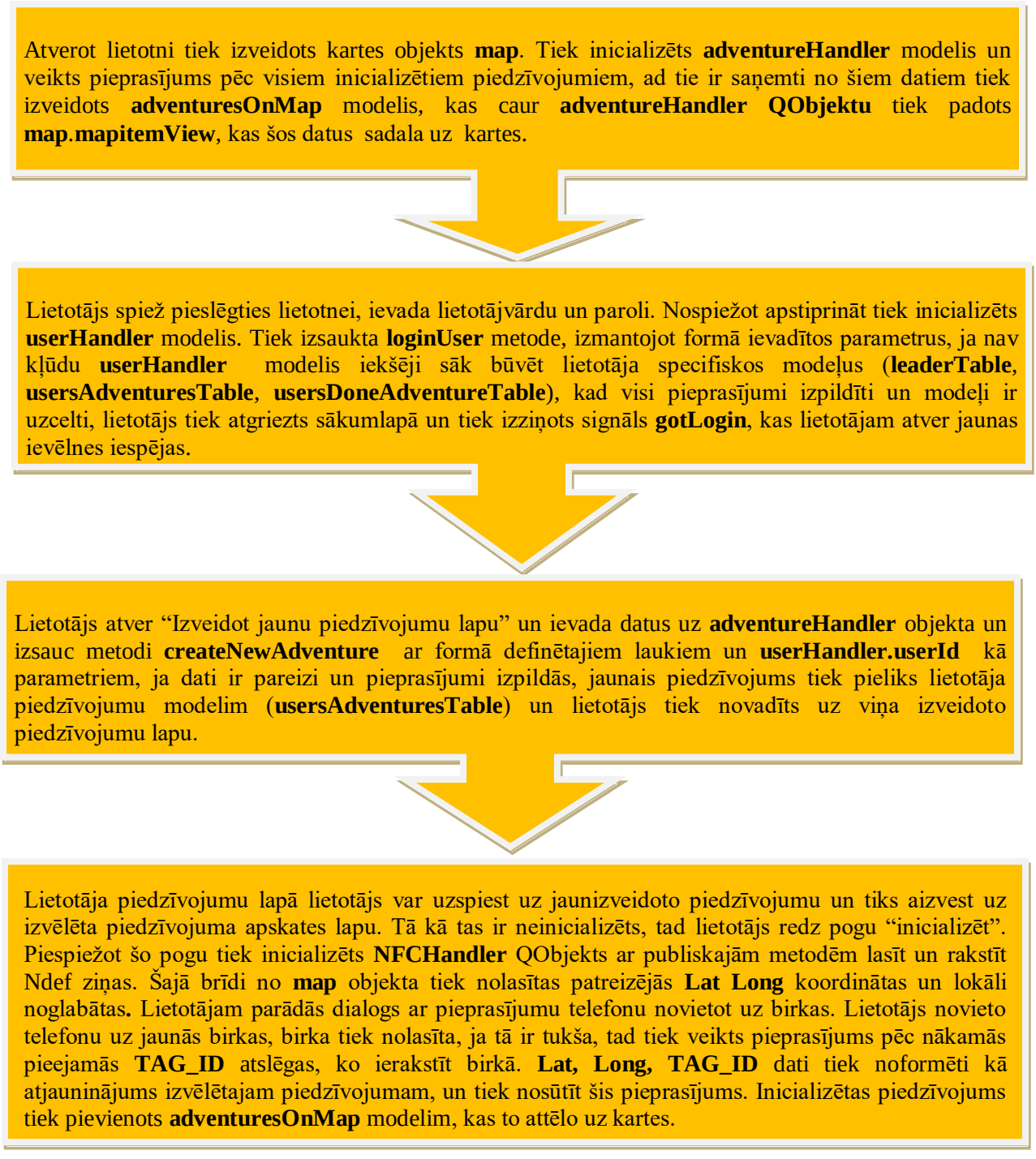


att. 06

5.3. Funkcionalitātes piemēri

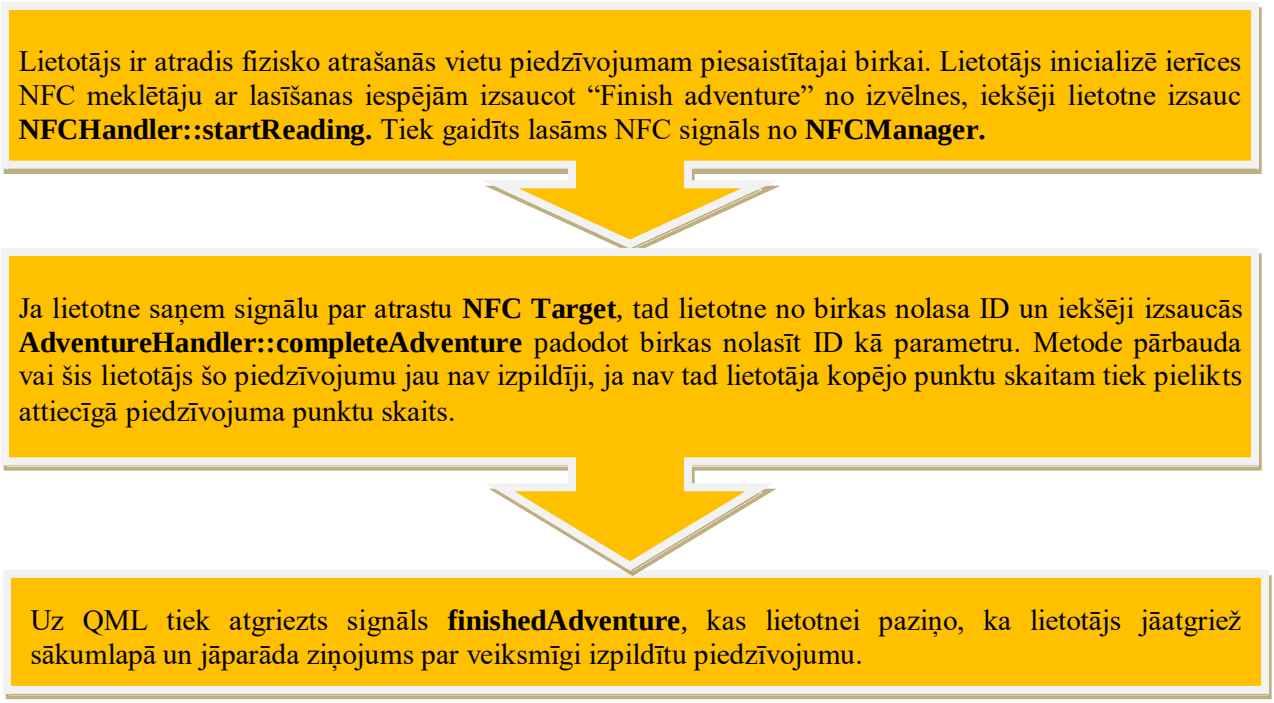
Labākai lietotnes izpratnei, tiek aprakstīts detalizēts funkcionalitātes piemērs, kurā tiek aprakstīts pilns piedzīvojuma izveides process. Programmas scenārijā iekļauti lietotāju darbības un raksturojumi lietotnes iekšējajiem procesiem.

5.3.1. Izveidot jaunu piedzīvojumu un to inicializēt



5.3.2. Izpildīt piedzīvojumu

Pirmie divi soļi sakrīt ar scenāriju - aprakstītu 4.3.1



5.4. Lietotnes funkcionālais projektējums

5.4.1. Klase UserHandler

Manto QObject. Klase ir atbildīga par visām ar lietotāju saistītajām funkcijām, kā arī uzbūvē visas ar lietotāju saistītās datu struktūras. Realizēts kā QObject un QML pieejams importējot NFCUser 0.1.

5.4.1.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	userId	+	+	+	Lietotāja Id
QString	login	+	+		Lietotājvārds

QString	email	+	+		E-pasts
QString	password	+	+		Parole
int	points	+	+	+	Punkti
int	role	+	+		Loma
QString	errorString		+		Kļūdas paziņojums
int	place		+		Vieta
QList<QObject *>	leaderTable		+		Līderu tabula
QList<QObject *>	usersAdventures Table		+	+	Lietotāju izveidotie piedzīvojumi
QList<QObject *>	usersDoneAdvent uresTable		+	+	Lietotāja izpildītie piedzīvojumu

5.4.1.2 Publiskās, no QML izsaucamās metodes:

Tips	**Metode**	**Apraksts**
Void	createNewUser	Nolasa uz QObject ievadītos datus, validē tos pēc nepieciešamības, ja dati pareizi izveido jaunu lietotāju. Pēc veiksmīgas izveides, izsauc getUserData un izziņo gotLogin signālu.
Void	loginUser	Parametri - QString p_login(lietotājvārds), QString p_pass (lietotāja parole). Pēc parametra datiem pārbauda vai eksistē lietotājvārda un paroles kombinācija, ja tāda atrasta, izsauc getUserData un izziņo gotLogin signālu.

Bool	getUserData	Parametri - QString p_login(lietotājvārds). Pēc parametra veic pieprasījumu, lai savāktu lietotājvārdu, lietotāja ID, e-pastu, punktus, lomu un vietu. Atgriež TRUE, ja lietotāja savākšana izdevusies.
Void	addAdventureToList	Parametri- int p_adventureId(piedzīvojuma Id), QString p_name (piedzīvojuma vārds), int p_award(balva), int p_status(status), QString p_desc(apraksts), QString p_clue(pavediens). Pievieno esošajai usersAdventuresTable jaunu, tikko izveidotu ierakstu, lai visa tabula nebūtu jāceļ pa jaunu.

5.4.1.3 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	userIdChanged		Izsaucās, kad Lietotāja ID ir mainījies.
Void	pointsChanged		Izsaucās, kad lietotāja punkti mainījušies.
Void	leaderTableChanged		Izsaucās, kad līderu tabulas dati mainījušies
Void	usersAdventuresTableChanged		Izsaucās, kad lietotāju piedzīvojumu tabulas dati mainījušies.
Void	usersDoneAdventureTableChanged		Izsaucās, kad lietotāju izpildīto piedzīvojumu tabulas dati mainījušies.
Void	gotLogin	Izsauc sprūdus:	Izsaucās, kad lietotājam

		buildLeaderboard buildUsersAdventureTable buildUsersDoneAdventureTable	veiksmīgi izdevies pieslēgties. Izsauc sprūdus.
Void	gotError	Izsauc sprūdu: handleError	Izsaucās, kad darbības gaitā ir notikusi kļūda. Izsauc sprūdu.
Void	error		Izsaucās no handleError, paredzēts, lai paziņotu QML , ka notikusi kļūda.
Void	startLoading		Izsaucās, kad sākas pieprasījumi uz ārējiem resursiem.
Void	endLoading		Izsaucās, kad beidzās pieprasījumi uz ārējiem resursiem.

5.4.1.4 Sprūdas:

Tips	**Signāls**	**Saistīts**	**Apraksts**
Void	handleError	Savienots ar signālu gotError.	Nostrādā errorString, ja tas mainījies, tad izziņo error signālu.
Void	buildLeaderboard	Savienots ar signālu gotLogin.	Uzceļ leaderTable.
Void	buildUsersAdventureTable	Savienots ar signālu gotLogin.	Uzceļ usersAdventuresTable.
Void	buildUsersDoneAdventureTable	Savienots ar signālu gotLogin.	Uzceļ usersDoneAdventuresTable.

5.4.1.5 Privātie mainīgie:

Tips	Nosaukums	Apraksts
int	l_userId	Lietotāja Id
QString	l_login	Lietotāj vārds
QString	l_email	Lietotāja epasts
QString	l_password	Lietotāja parole
int	l_points	Lietotāja punkti
int	l_role	Lietotāja loma
int	l_place	Lietotāja vieta
QString	l_error	Kļūdas paziņojums
QSqlDatabase	l_db	Datubāzes savienojums
QList<QObject *>	l_leaderTable	Līderu tabula
QList<QObject *>	l_userAdventureTable	Lietotāja piedzīvojumu tabula
QList<QObject *>	l_userDoneAdventureTable	Lietotāja izpildīto piedzīvojumu tabula
bool	l_loading	Vai šobrīd notiek pieprasījums uz ārējiem resursiem

5.4.2. Klase AdventureHandler

Manto QObject. Klase ir atbildīga par piedzīvojumiem saistītajām funkcijām, lai to izmantotu QML jāimportē NFCAdventure 0.1.

5.4.2.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	adventureId	+	+	+	Piedzīvojuma ID.
int	ownerId	+	+		Piedzīvojuma īpašnieka ID.
int	tagId	+	+	+	Birkas ID.
QString	name	+	+	+	Piedzīvojuma nosaukums.
QString	desc	+	+	+	Piedzīvojuma apraksts.
QString	clue	+	+	+	Piedzīvojuma loma.
int	award	+	+	+	Piedzīvojuma atalgojums.
double	geoLat	+	+	+	Koordinātas platumu grādu komponente.
double	geoLong	+	+	+	Koordinātas garumu grādu komponente.
QString	errorString		+	+	Kļūdas paziņojums.
int	status	+	+	+	Patreizējas piedzīvojuma status.
QList<QObject *>	adventuresOnMap		+	+	Modelis ar inicializētajiem piedzīvojumiem.

5.4.2.2 Publiskās, no QML izsaucamās metodes:

Tips	Metode	Apraksts
Void	createNewAdventure	Parametri - int ownerId(lietotāja ID, kas veic pieprasījumu). Nolasa uz QObject ievadītos datus, validē tos pēc nepieciešamības, ja dati pareizi izveido jaunu piedzīvojumu. Pēc veiksmīgas izveides izziņo gotAdventure signālu.
Void	getAdventureData	Parametri - int p_adventureId (Piedzīvojuma ID). Pēc parametra datiem pārbauda vai eksistē šāds piedzīvojums, ja eksistē, sapilda objekta datus ar šī piedzīvojuma raksturdatiem.
Void	initAdventure	Parametri - int p_adventureId (Piedzīvojuma ID), int p_tagId (Birkas ID), p_lat (platuma koordināta), p_long (garuma koordināta) Pēc parametra datiem pārbauda vai eksistē šāds piedzīvojums un vai šīs birkas ID ir unikāls, ja kļūdu nav atjaunina šo piedzīvojuma ierakstu un pievieno birkasID, Lat un Long datus. Pēc veiksmīgas izveides izziņo gotInit signālu.
Void	completeAdventure	Parametri - int p_adventureId (Piedzīvojuma ID), int p_tagId (Birkas ID), p_userId (Izpildītāja ID). Pēc parametra datiem pārbauda vai eksistē šāds piedzīvojums un vai tam pieder šāds birkas ID, ja dati sakrīt, tad lietotāja kopējo punktu skaitam tiek pieskaitīti attiecīgā piedzīvojuma punkti. Pēc veiksmīgas transakcijas izziņo completeAdventure signālu.
Void	buildAdventuresOnMap	Pieprasa visus adventuresOnMap module, kas tiek padots

		QML lai tas tiktu izdalīts pa karti.
Void	deleteAdventure	Parametri - int p_adventureId (Piedzīvojuma ID). Pēc parametra datiem pārbauda vai šim lietotājam eksistē šāds piedzīvojums, ja eksistē, tad tiek izdzēsti visi izpildīto piedzīvojumu ieraksti ar šo piedzīvojum un pats šis piedzīvojums. Punkti netiek noņemti lietotājiem, kas šo piedzīvojumu ir izpildījuši

5.4.2.3 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	adventureIdChanged		Izsaucās, kad piedzīvojuma ID ir mainījies.
Void	tagIdChanged		Izsaucās, kad birkas Id mainījies.
Void	nameChanged		Izsaucās, kad piedzīvojuma nosaukums mainījies.
Void	descChanged		Izsaucās, kad piedzīvojuma apraksts mainījies.
Void	clueChanged		Izsaucās, kad piedzīvojuma pavediens mainījies.
Void	awardChanged		Izsaucās, kad piedzīvojuma atalgojums mainījies.
Void	geoLatChanged		Izsaucās, kad piedzīvojuma platuma komponente mainījusies.
Void	geoLongChanged		Izsaucās, kad piedzīvojuma

			garuma komponente mainījusies.
Void	statusChanged		Izsaucās, kad piedzīvojuma status ir mainījies.
Void	gotError	Izsauc sprūdu: handleError	Izsaucās, kad darbības gaitā ir notikusi kļūda. Izsauc sprūdu.
Void	gotAdventure	Izsauc sprūdu: addAdventureOnMap	Izsaucās, kad ir izveidots jauns piedzīvojums
Void	completedAdventure		Izsaucās, kad pabeigts piedzīvojums
Void	initAdvnture		Izsaucās, kad veiksmīgi inicializēts piedzīvojums.
Void	error		Izsaucās no handleError, paredzēts, lai paziņotu QML, ka notikusi kļūda.
Void	errorChanged		Izsaucās, kad kļūdas paziņojums ir mainījies.
Void	startLoading		Izsaucās, kad sākas pieprasījumi uz ārējiem resursiem.
Void	endLoading		Izsaucās, kad beidzās pieprasījumi uz ārējiem resursiem.

5.4.2.4 Sprūdas:

Tips	Signāls	Saistīts	Apraksts
Void	handleError	Savienots ar signālu gotError.	Nostrādā goError, ja tas mainījies tad izziņo error signālu.
Void	addAdventureOnMap	Savienots ar signālu initAdventure.	Pievieno piedzīvojumu adventuresOnMap modelim, lai tas viss nebūtu jāpārlādē vēlreiz.

5.4.2.5 Privātie mainīgie:

Tips	Nosaukums	Apraksts
int	l_adventureId	Piedzīvojuma Id.
int	l_ownerId	Piedzīvojuma īpašnieka ID.
int	l_tagId	Birkas ID.
QString	l_name	Piedzīvojuma vārds.
QString	l_desc	Piedzīvojuma apraksts.
QString	l_clue	Piedzīvojuma pavediens.
int	l_award	Piedzīvojuma atalgojums.
int	l_status	Piedzīvojuma status.
double	l_geoLat	Piedzīvojuma koordinātas platuma komponente.
double	l_geoLong	Piedzīvojuma koordinātas garuma komponente.
QList<QObject *>	l_adventuresOnMap	Objektu saraksts, izmantots kā modelis, lai izkārtotu

		inicializētus piedzīvojumus kartē.
QString	l_error	Kļūdas paziņojums.
QSqlDatabase	l_db	Datubāzes savienojums.
bool	l_loading	Vai šobrīd notiek pieprasījums uz ārējiem resursiem.

5.4.3. Klase AdventureOnUserData

Manto QObject. Klase paredzēta, lai veidotu QML interpretējamu QObject sarakstu, kas tiek padots kā modelis. Klasi izmanto, lai būvētu lietotāja izveidotos piedzīvojumus un lietotāja izpildītos piedzīvojumus.

5.4.3.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	adventureId	+	+	+	Piedzīvojuma ID.
QString	name	+	+	+	Piedzīvojuma nosaukums.
int	award	+	+	+	Piedzīvojuma atalgojums.
int	status	+	+	+	Piedzīvojuma status.
QString	desc	+	+	+	Piedzīvojuma apraksts.
QString	clue	+	+	+	Piedzīvojuma pavediens.

5.4.3.2 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	adventureIdChanged		Izsaucās, kad piedzīvojuma ID ir mainījies.

Void	nameChanged		Izsaucās, kad piedzīvojuma nosaukums mainījies.
Void	descChanged		Izsaucās, kad piedzīvojuma apraksts mainījies.
Void	clueChanged		Izsaucās, kad piedzīvojuma pavediens mainījies.
Void	awardChanged		Izsaucās, kad piedzīvojuma atalgojums mainījies.
Void	statusChanged		Izsaucās, kad piedzīvojuma status ir mainījies.

5.4.3.3 Privātie mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	l_adventureId	Piedzīvojuma Id.
QString	l_name	Piedzīvojuma vārds.
QString	l_desc	Piedzīvojuma apraksts.
QString	l_clue	Piedzīvojuma pavediens.
int	l_status	Piedzīvojuma status.

5.4.4. Klase LeaderboardData

Manto QObject. Klase paredzēta, lai veidotu QML interpretējamu QObject sarakstu, kas tiek padots kā modelis. Klasi izmanto lai būvētu Līderu sarakstu.

5.4.4.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	place	+	+	+	Lietotāja vieta sarakstā.
QString	login	+	+	+	Lietotāja lietotājvārds.
int	points	+	+	+	Lietotāja punkti.

5.4.4.2 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	placeChanged		Izsaucās, vieta ir mainījusies.
Void	loginChanged		Izsaucās, kad lietotājvārds ir mainījies.
Void	pointsChanged		Izsaucās, kad piedzīvojuma punkti ir mainījušies.

5.4.4.3 Privātie mainīgie:

Tips	Nosaukums	Apraksts
int	l_place	Lietotāja vieta.
QString	l_login	Lietotāja lietotājvārds.
l_points	l_points	Leitotāja kopējais punktu skaits.

5.4.5. Klase MapItemData

Manto QObject. Klase paredzēta, lai veidotu QML interpretējamu QObject sarakstu, kas tiek padots kā modelis. Klasi izmanto, lai būvētu piedzīvojumus, kurus attēlot kartē ka punktus.

5.4.5.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	adventureId	+	+	+	Piedzīvojuma ID.
int	tagId	+	+	+	Birkas ID.
QString	name	+	+	+	Piedzīvojuma nosaukums.
double	geoLat	+	+	+	Piedzīvojuma koordinātas garuma komponente.
double	geoLong	+	+	+	Piedzīvojuma koordinātas platuma komponente
QString	award	+	+	+	Piedzīvojuma atalgojums.

5.4.5.2 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	adventureIdChanged		Izsaucās, kad piedzīvojuma ID ir mainījies.
Void	tagIdChanged		Izsaucās, kad birkas ID ir mainījies.
Void	nameChanged		Izsaucās, kad piedzīvojuma nosaukums ir mainījušies.

Void	geoLatChanged		Izsaucās, kad piedzīvojuma koordinātas platuma komponente ir mainījusies.
Void	geoLongChanged		Izsaucās, kad iedzīvojuma koordinātas garuma komponente ir mainījusies.
Void	awardChanged		Izsaucās, kad piedzīvojuma atalgojums ir mainījies.

5.4.5.3 Privātie mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	l_adventureId	Piedzīvojuma ID.
int	l_tagId	Birkas ID.
QString	l_name	Piedzīvojuma nosaukums.
double	l_geoLat	Piedzīvojuma koordinātas garuma komponente.
double	l_geoLong	Piedzīvojuma koordinātas platuma komponente.
QString	l_award	Piedzīvojuma atalgojums.

5.4.6. Klase System

Manto QObject. Klase paredzēta, lai veidotu QML interpretējamu QObject. Klase paredzēta ar risinājumiem saistītiem ar sistēmu un izpildes vidi.

5.4.6.1 Publiskās, no QML izsaucamās metodes:

Tips	Metode	Apraksts
Qstring	getEnv	Atgriež pašreizējā kernel tipu. Tiek izmantots, lai mainītu saskarni starp dažādam ierīcēm.

5.4.7. Klase NfcDb

Klase paredzēta, lai viens datubāzes savienojums būtu pieejams visur sistēmā.

5.4.7.1 Publiskās, no QML izsaucamās metodes:

Tips	Metode	Apraksts
QSqlDatabse	getDb	Ja datubāzes savienojums nav izveidots, tad izveido jaunu un atgriež to, bet, ja tas ir izveidots, atgriež esošo savienojumu.

Privātie mainīgie:

Tips	Nosaukums	Apraksts
QSqldatabse	l_db	Datubāzes savienojums.

5.4.8. Klase NfcHandler

Manto QObject. Klase ir atbildīga par birkām saistītajām funkcijām, Lai to izmantotu QML jāimportē NFCHandler 0.1.

5.4.8.1 Raksturdati kā Q_PROPERTY:

Tips	Nosaukums	Rakstīt	Lasīt	Ir mainīts signāls	Apraksts
int	tagId	+	+	+	Birkas ID.

QNdefNfcTextRecord	nfcText	+	+	+	NDEF formāta teksta ieraksts.
QNdefMessage	nfcMessage	+	+	+	NDEF formāta teksta ieraksts.
QString	errorString		+	+	Kļūdas paziņojums.

Stadijas(Action):

Tips	**Apraksts**
NoAction	setTargetAccessModes tiek uzstādīts kā NoAccess, kas nozīmē, ka lietotne neveiks nekādas darbības ar uztvertiem NFC signāliem.
ReadNdef	Inicializēts no startReading. setTargetAccessModes tiek uzstādīts kā NdefReadTargetAccess, kas nozīmē, ka lietotne meklē NFC uztvērēju ar rakstīšanas atļauju.
WriteNdef	Inicializēts no startWriting. setTargetAccessModes tiek uzstādīts kā NdefWriteTargetAccess, kas nozīmē, ka lietotne meklē NFC uztvērēju ar lasīšanas atļauju.

5.4.8.2 Publiskās, no QML izsaucamās metodes:

Tips	**Metode**	**Apraksts**
Void	startReading	Uzstāda stadiju, kā ReadNdef, uzstāda pieejas režīmu lasīšanai un sāk meklēt mērķi.
Void	startWriting	Uzstāda stadiju, kā WriteNdef, uzstāda pieejas režīmu rakstīšanai un sāk meklēt mērķi.
Void	stopLooking	Uzstāda stadiju kā NoAction un beidz meklēt mērķus.

Void	addText	Parametri - QNdefNfcTextRecord textRecord (pievienojamai teksta ieraksts). Pievieno NFC ziņai jaunu teksta ierakstu.

5.4.8.3 Signāli:

Tips	Signāls	Saistīts	Apraksts
Void	tagIdChanged		Izsaucās, kad birkas Id mainījies.
Void	nfcTextChanged		Izsaucās, kad piedzīvojuma nosaukums mainījies.
void	nfcMessageChanged		
Void	gotError	Izsauc sprūdu: handleError	Izsaucās, kad darbības gaitā ir notikusi kļūda. Izsauc sprūdu.
void	error		Izsaucās no handleError, lai paziņotu QML, ka notikusi kļūda.

5.4.8.4 Sprūdas:

Tips	Signāls	Saistīts	Apraksts
Void	targetDetected	NfcManager::targetDetected	Parametri - QNdefMessage &message (Ziņa no NfcManager), QNearFieldTarget *target(norāde uz atrasto mērķi). Nostrādā tiklīdz atrasts

			atbilstošs NFC signāls, norāde uz šo birku tiek padota caur parametru. Iekšēji, pēc klases stadijas veic attiecīgo darbību (lasīt/rakstīt).
Void	targetLost	NfcManager::targetLost	Parametri - QNearFieldTarget *target(norāde uz atrasto mērķi). Tiek izdzēsts mērķis.
Void	targetError	NfcManager::targetError	Parametri - QNearFieldTarget::Error error (atgrieztā kļūda), const QNearFieldTarget::RequestId &id(pieprasījuma ID). Tiek nostrādāti dažādās iespējamās atgrieztās kļūdas un ar gotError tās tiek signalizētas.
Void	handleError	Savienots ar signālu gotError	Nostrādā goError, ja tas mainījies, tad izziņo error signālu.

5.4.8.5 Privātie mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	l_tagId	Birkas Id.
QString	l_message	Ziņa Qstring formātā.
QString	l_nfcMessage	Ziņa pilnā NDEF formātā.

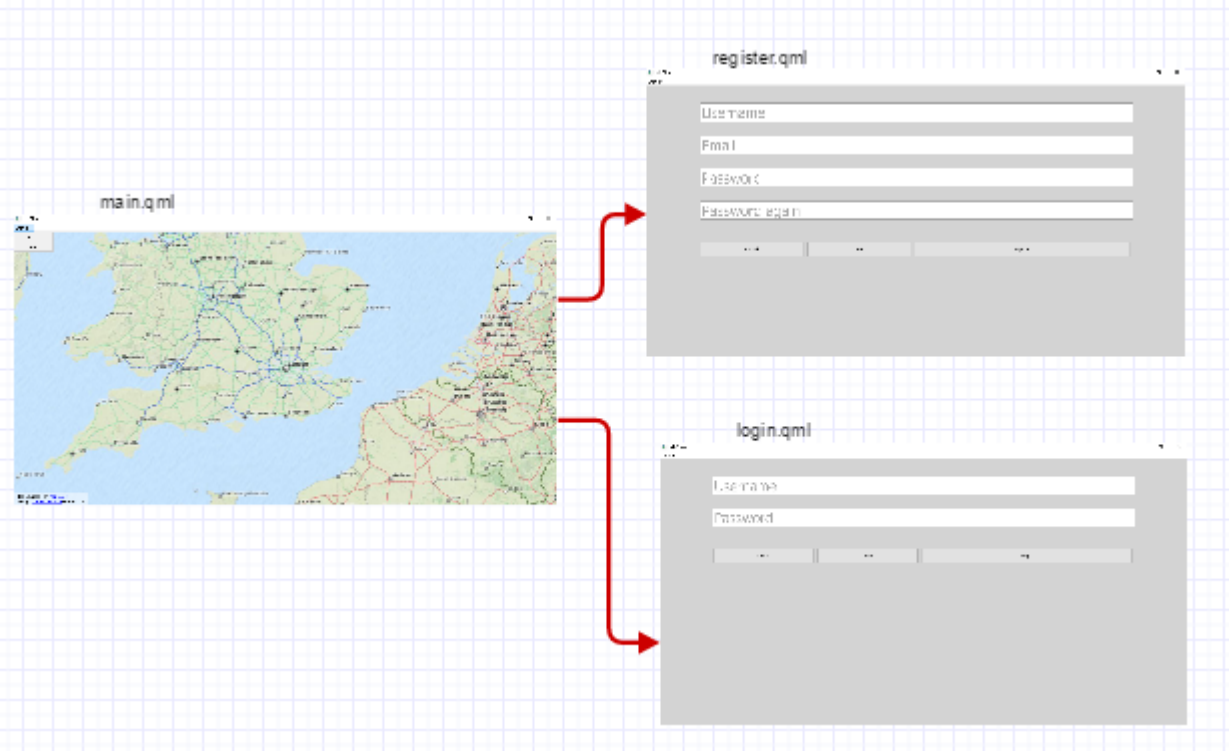
QNdefMessage	l_nfcText	Teksta daļa no ziņas NDEF formātā.
QNdefNfcTextRecord	l_manager	Norāde uz NFC manager.
QNearFieldManager	l_target	Norāde uz NFC target.
QNearFieldTarget::RequestId	l_request	NFC pieprasījums.
Action	l_action	Klases stadijas.

5.5. Skatu izkārtojumi

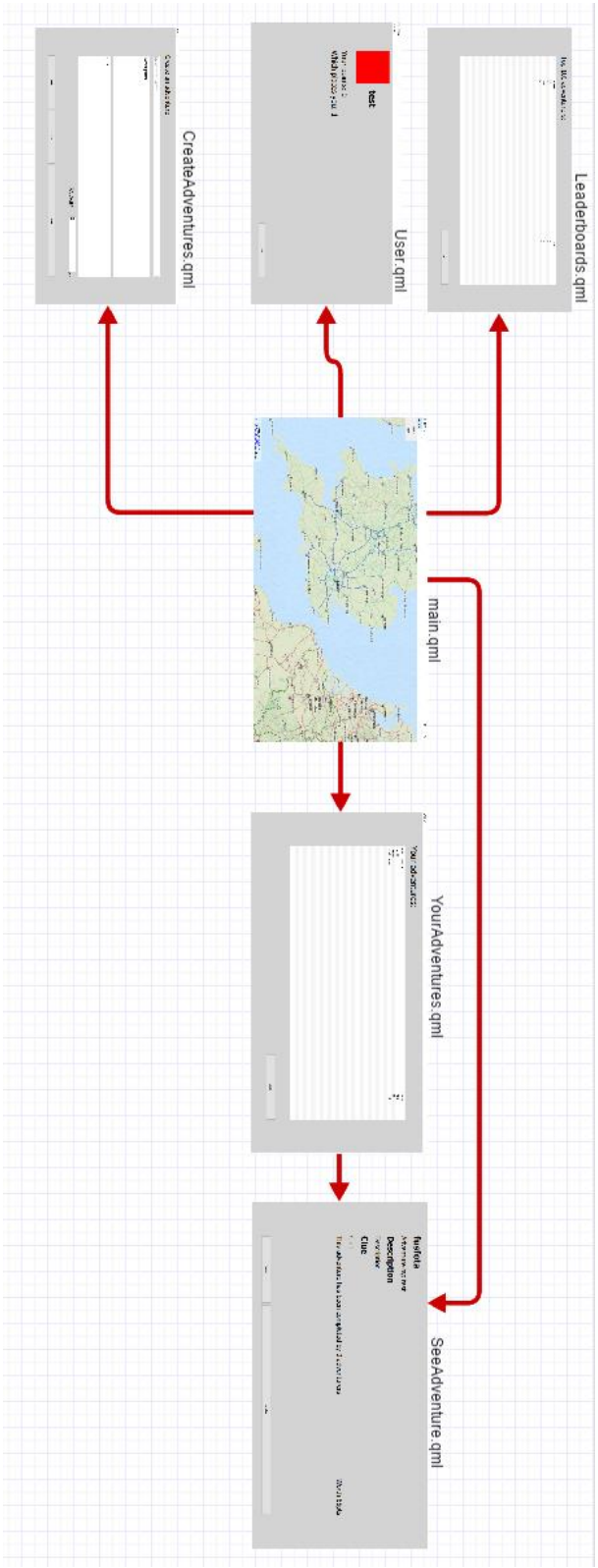
Nodaļa tiks aprakstītas izveidotās formas un objekti, kas izmantoti saskarnes veidošanai. Qt pamatobjekti netiks aprakstīti.

5.5.1. Saskarņu diagrammas

5.5.1.1 Neautentificētā lietotāja skati:



5.5.1.2 Autentificēta lietotāja skati



5.5.2. Main.qml

Galvenais skats sevī ietver visus pārējos skatus.

5.5.2.1 Bērni

Id	Forma/objekts
mainMenu	MainMenu.qml
mapPopup	MapPopup.qml
mapComponent	MapComponent.qml
mainUserHandle	HandleUser(QObject)
thisAdvendture	HandleAdventure(QObject)
thisSystem	System(QObject)

5.5.2.2 Skata metodes:

Metode	Apraksts
createMap	Metode atverot aplikāciju uzceļ karti un uz tās izvieto modeli adventuresOnMap.

5.5.3. MainMenu.qml

Galvenā izvēlne, kas sevī ietver izsaukumus uz dažādiem citiem skatiem atkarībā no tā vai lietotājs ir pieslēdzies.

5.5.3.1 Parametri

Tips	Nosaukums	Apraksts
alias	actionMenu	Norāde uz bērnu Menu, lai vecāks varētu tikt pie tā datiem.

5.5.3.2 Signāli

Nosaukums	Apraksts
selectAction	Parametri – string action (izvēlētā sadaļa izvēlnē). Signāls, ko vecāks uztver, lai stackView nosūtītu attiecīgi izsaukto lapu.

5.5.3.3 Skata metodes:

Metode	Apraksts
createMenu	Parametri – bool isLogin Pēc padotā parametra izvēlas kāda veida izvēlni celt.

5.5.4. MapPopup.qml

Neliela izvēlne, kas tiek izsaukta pēc konkrēta signāla, šajā gadījumā, pēc tam, kad uz kartes turēta nospiesta kreisā peles poga.

5.5.4.1 Parametri

Tips	Nosaukums	Apraksts
variant	coordinate	Pēdējā klikšķa koordināta.

5.5.4.2 Signāli

Nosaukums	Apraksts
itemClicked	Parametri – string item (izvēlētais rīks izvēlnē). Signāls, ko vecāks uztver, lai stackView nosūtītu attiecīgi izsaukto lapu.

5.5.4.3 Skata metodes:

Metode	Apraksts
update	Izveido izlecošo izvēlni.

5.5.1. HandleUser(mainUserHandle)

Galvenais lietotnes lietotājs, tiek aizpildīts un apstrādātos no userHandler.h.

5.5.1.1 Parametri

Tips	Nosaukums	Apraksts
bool	userOK	Vai lietotājs ir pieslēdzies.

bool	loading	Vai šobrīd notiek kādi pieprasījumi uz ārējiem resursiem.

5.5.1.2 Signāli

Nosaukums	**Apraksts**
onError	Izmet dialoga paziņojumu ar kļūdas paziņojumu no objekta.
onGotLogin	Pēc pieslēgšanas pārbūvē izvēlnes iespējas lietotājam.

5.5.2. HandleAdventure (thisAdvendture)

Objekts tur aktuālā piedzīvojuma pilnos datus un tā metodes.

5.5.2.1 Signāli

Nosaukums	**Apraksts**
onError	Izmet dialoga paziņojumu ar kļūdas paziņojumu no objekta.
onGotAdventure	Paziņo, ka veiksmīgi izveidots piedzīvojums.

5.5.3. System (thisSystem)

Objekts satur metodes, kas ļauj nolasīt sistēmas/platformas raksturdatus.

5.5.4. CreateAdventures.qml

Skats jauna piedzīvojuma izveidošanai.

5.5.4.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.
createdAdventure	Paziņo vecākam, ka veiksmīgi izveidots piedzīvojums.

5.5.5. Login.qml

Skats lietotāja pieslēgšanai.

5.5.5.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.
gotLogin	Paziņo vecākam, ka lietotājs pieslēdzies.

5.5.6. Register.qml

Skats lietotāja reģistrēšanai.

5.5.6.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.
gotLogin	Paziņo vecākam, ka lietotājs pieslēdzies.

5.5.7. User.qml

Attēlo izvēlētā lietotāja raksturdatus.

5.5.7.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.
gotLogin	Paziņo vecākam, ka lietotājs pieslēdzies.

5.5.7.2 Parametri

Tips	Nosaukums	Apraksts
string	userLogin	Litotājvārds
string	userPlace	Lietotāja vieta
String	userPoints	Lietotāja punkti

5.5.8. Leaderboard.qml

Skats līderu tabulas attēlošanai

5.5.8.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.

5.5.9. SeeAdventure.qml

Attēlo izvēlētā lietotāja raksturdatus.

5.5.9.1 Signāli

Nosaukums	Apraksts
closeForm	Paziņo StackView, ka šī lapa jāaizver.

5.5.9.2 Parametri

Tips	Nosaukums	Apraksts
string	ownerUserLogin	Īpašnieka litotājvārds.
int	thisAdventureId	Izvēlētā piedzīvojuma ID.
string	thisAdventureName	Izvēlētā piedzīvojuma nosaukums.
string	thisAdventureDesc	Izvēlētā piedzīvojuma apraksts.
string	thisAdventureClue	Izvēlētā piedzīvojuma pavediens.
int	thisAdventureReward	Izvēlētā piedzīvojuma balva
int	thisAdventureCompletedBy	Skaits, cik lietotāju šo piedzīvojumu izpildījuši.
Bool	thisAdventureInit	Vai šis piedzīvojums ir inicializēts.

6. Projekta organizācija

Projektā tika izvēlēta spirāles modeļa programmēšanas pieeja, jo pilnā produkta perspektīva ir plašāka un nevarētu tikt ietverta kvalifikācijas darba projekta ietvaros. Programmprodukta izstrāde notika ar pakāpenisku izstrādi, tika izstrādāti atsevišķie modeļi, lai nodrošinātu pamata funkcionalitāti, tos testējot ar rokas ievadi.

Pakāpeniski pievienojot jaunu funkcionalitāti produktam tika veidoti arī aizbāžņi un pagaidu risinājumi ērtākai datu ievadei.

Programmatūras izstrāde notika balstoties uz mazliet atvasinātu Geotechnical Software Services C++ Programming Style Guidelines[1].

Izstrādes progresa vieglākai sekošanai tika izveidota trello kanban tāfele[2], kas ļauj viegli plānot un sekot uzdevumu norisei, uzskatāmi tos vizualizējot ar dažādām stadijās izstrādes gaitā. Projektā tika pielietotas 5 stadijas:

- Jāizdara
- Progresā
- Koda pārskats
- Vienībtesti
- Izpildīts

6.1. Konfigurāciju pārvaldība

Konfigurāciju pārvaldībai un versiju kontrolei tika izmantots Git, projekta publiskā krātuve atrodama https://github.com/HermanisV/NFCLUES/.

Jauni un eksperimentāli risinājumi tika veidoti atsevišķos atzaros, kas, ja tika pieņemti, tika vēlāk iekļauti programmatūras konfigurācijas pamatstumbrā.

[1] Saite - http://geosoft.no/development/cppstyle.html [skatīts 26.05.2016]

[2] Saite - https://trello.com/nfclues

6.2. Kvalitātes nodrošināšanā

Lai nodrošinātu kvalitāti, tika piekopta šāda prakse:

- Izstrāde tika vadīta pēc objektorientētās programmēšanas principiem.
- Sarežģītāki risinājumi, kā arī funkciju un procedūru prototipi, izstrādes gaitā tika skaidroti ar komentāriem.
- Lietotnes vienībtesti tika veikti gan uzreiz pēc jaunās funkcionalitātes izveides, gan arī turpmāko citu funkciju vienībtestos.
- Koda noformēšanai tika izmantots Qt Creator iekšējais uzkopējs.
- Pēc jaunas funkcionalitātes pievienošanas, risinājums tika pievainots vietējajai krātuvei, un vēlāk publicēts publiskajā krātuvē.

7. Darbietilpības novērtējums

Darbietilpības novērtējums tika izstrādāts projekta beigās, ņemot vērā līdz novērtēšanas brīdim izstrādāto risinājumu.

Ņemot apvienotos QML un C++ risinājums, izstrādātā darba koda rindiņu skaits pārsniedz 3000. Darba izstrādātājs bija pazīstams ar C++ un JavaScript valodu nominālā līmenī. Darbietilpības novērtējumā jāņem vērā izstrādātāja QML tehnoloģijas nepazīšana, kā arī QML un C++ savstarpējā sajūgšana, kā arī izstrādātājam bija jāiepazīstas ar QT ietvara bibliotēkām, sniegtajām funkcijām un risinājumiem.

Ņemot vērā šos nosacījumus COCOMO II cenas modelis projekta darbietilpību aprēķināja, kā 8 mēnešus, kas diezgan droši ir pārāk liels novērtējums, visdrīzākais tāpēc, ka QML ļoti script-veida deklaratīva valoda, kas rezultējās uz lielu rindiņu skaitu.

Pēc personīgas analīzes šo projektu varētu novērtēt no 4 līdz 5 mēnešiem, ņemot vērā izstrādātāja iepriekšējās zināšanas un sagatavotību.

8. Testēšanas dokumentācija

Testēšana noritēja gan izstrādes gaitā testējot jaunu risinājumu vienumus, gan veicot programmprodukta kopējo gatavības testu. Tie noritēja izmantojot tiešo ievadi un novērojot rezultātus izvadē, kā arī analizējot lietotnes izvadi uz konsoli.

8.1. Testēšanas scenāriji

8.1.1. Jebkurš var izveidot jaunu lietotāju

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver reģistrēšanās skatu.	Atveras skats.		+
Lietotājs ievada validācijām atbilstošus datus un spiež "Apstiprināt".	Lietotājs veiksmīgi reģistrējās.	Punktu skaits pēc reģistrēšanas lietotājam saglabāts nepareizs.	+
		Pēc pieslēgšanās lietotājs netiek novirzīts uz sākumlapu.	+
		Lietotāja vieta neparādās.	+
Lietotāja parole un parole atkārtoti nesakrīt.	Parādās kļūdas paziņojums.		+
Ievadītais lietotājvārds ir par īsu.	Parādās kļūdas paziņojums.		+
Ievadītā parole ir par īsu.	Parādās kļūdas paziņojums.		+
Lietotāja ievadītais lietotājvārds nav unikāls.	Parādās kļūdas paziņojums.	Kļūdas paziņojums parādās tikai vienreiz.	+

8.1.2. Lietotājs var pieslēgties ar esošu lietotāju

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver pieslēgšanās skatu.	Atveras skats.		+
Lietotāja ievadītais lietotājvārds netiek atrasts.	Parādās kļūdas paziņojums.	Kļūdas paziņojums paradās tikai vienreiz.	+
Lietotājs ievada paroli, kas nesakrīt ar paroli zem šī lietotājvārda.	Parādās kļūdas paziņojums.		+
Lietotājs ievada atbilstošu paroli un lietotājvārdu un spiež "Apstiprināt".	Lietotājs veiksmīgi pieslēdzās.	Kļūdas paziņojums parādās pat, ja paroles sakrīt.	+

8.1.3. Lietotājs apskata savu profilu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver profila skatu.	Atveras skats un tiek norādīti lietotāja raksturdati.	Lietotāja punktu skaits rādās nepareizs.	+
		Lietotāja lietotājvārds ir tukšs.	+
		Lietotāja vieta ir tukša.	+

8.1.4. Lietotājs apskata savus piedzīvojumus

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver viņa piedzīvojumu skatu.	Atveras skats un tabulā tiek uzskaitīt lietotāja piedzīvojumi.	Lietotāja piedzīvojumi netiek atgriezti.	+
		Pēc jauna piedzīvojuma izveides tas neparādās piedzīvojumu sarakstā.	+
Lietotājs uzspiež uz piedzīvojuma.	Lietotājs tiek novests uz ”apskatīt piedzīvojumu” skatu.	Pēc novešanas uz jauno skatu, atgriešanās aizved uz sākumlapu, skatīt piedzīvojumus lapu.	+

8.1.5. Lietotājs apskata Līderu tabulu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver līderu tabulu.	Atveras skats ar aizpildītu līderu tabulu.	Jauns lietotājs neparādās tabulā tikko pēc izveides.	+

8.1.6. Lietotājs izveido jaunu piedzīvojumu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver jaunu piedzīvojumu izveides skatu.	Atveras skats ar jaunu piedzīvojumu izveides formu.		+
Lietotājs ievada validācijām atbilstošus datus un spiež “Apstiprināt”.	Tiek izveidots jauns neinicializēts piedzīvojums un lietotājs tiek novirzīts uz viņa piedzīvojumu lapu.	Lietotājs netiek novirzīts uz viņa piedzīvojumu lapu.	+
		Jaunajam piedzīvojumam nav statusa.	+
		Jaunais piedzīvojums neparādās lietotāja piedzīvojumu tabulā.	+
Ievadītais nosaukums sakrīt ar nosaukumu kāds jau ir šim lietotajam.	Parādās kļūdas paziņojums.		+

Nosaukums, apraksts, pavediens vai balva atstāti tukši.	Parādās kļūdas paziņojums.	Ja balva atstāta tukša, parādās neapstrādāts kļūdas paziņojums.	+

8.1.7. Lietotājs apskata konkrētu piedzīvojumu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs atver piedzīvojumu no savu piedzīvojumu saraksta.	Atveras skats ar aizpildītu šī piedzīvojuma raksturdatiem.	Poga "Inicializēt" ir pieejama pat inicializētiem piedzīvojumiem.	+
Lietotājs atver piedzīvojumu no izpildīto piedzīvojumu saraksta.	Atveras skats ar aizpildītu šī piedzīvojuma raksturdatiem.	Poga "Inicializēt" ir pieejama pat ja lietotājam šis piedzīvojums nepieder.	+
Lietotājs atver piedzīvojumu no kartes piedzīvojuma.	Atveras skats ar aizpildītu šī piedzīvojuma raksturdatiem.		29.05.2016 nav apstiprināts

8.1.8. Lietotājs izpilda piedzīvojumu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs ieslēdz ierīci NFC lasīšanas režīmā.	Paziņojums, ka ierīce meklē birku.	Ierīce sāk meklēt birku, bet neparāda paziņojumu.	+
Lietotāja ierīce atrod birku un to nolasa.	Paziņojums, ka birka atrasta, tiek attēlots birkas ID.	Neparādās paziņojums.	+
Lietotne atrod atbilstošo piedzīvojumu un tā punktu skaitu pieskaita lietotāja kopējajam punktu skaitam.	Lietotāja punktu skaits Palielinās par attiecīgo punktu daudzumu, un izpildītais piedzīvojums, izpildīto piedzīvojumu sarakstā.	Lietotāja punktu skaits neatjaunojas, līdz tas nepieslēdzas no jauna.	+
		Izpildītais piedzīvojums neparādās izpildīto piedzīvojumu sarakstā.	+

8.1.9. Lietotājs inicializē piedzīvojumu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs spiež pogu inicializēt, izvēlētā piedzīvojuma lapā.	Ierīce sāk meklēt NFC birku rakstīšanas režīmā.	Ierīce sāk meklēt birku, bet neparāda paziņojumu.	29.05.2016 nav apstiprināts

8.1.10. Lietotājs dzēš savu piedzīvojumu

Izpilde	Gaidāmais rezultāts	Dokumentētās problēmas	Atrisināts
Lietotājs spiež pogu "dzēst piedzīvojumu", izvēlētā piedzīvojuma lapā.	Vaicājuma paziņojums, vai lietotājs tiešam vēlas dzēst piedzīvojumu.		+
Lietotājs piekrīt dzēst piedzīvojumu	Tiek dzēsti visi ieraksti, no Done_adventure un Advnetures tabulām, kuriem ir šis piedzīvojuma ID.	Netiek izdzēsti ieraksti no Done_adventures tabulas.	+

9. Nobeigums

Pirmajā izstrādes ciklā izstrādātais programmprodukts nesasniedza visas paredzētās prasības, tās tika pārceltas uz nākamo izstrādes ciklu. Galvenais iemesls pilnu prasību nesasniegšanai bija izstrādātāja zināšanu trūkums par WEB servisiem un WEB pieprasījumiem. Pirmajā ciklā neizdevās izveidot pareizi funkcionējošu sajūgumu ar DreamFactory API, tāpēc tas pilnā formā netika iekļauts pirmā cikla galaproduktā. Sajūguma trūkums ar DreamFactory liedza izmantot pilnu programmprodukta funkcionalitāti uz Android ierīcēm.

Pirmajā iterācijā vairāki automatizēti risinājumi priekš Androīd ierīcēm tika aizvietoti ar aizbāžņiem priekš Windows vides. Sajūdzot šīs abas vides, Windows priekš datu ievades datubāzēs un Androīd priekš piedzīvojumu meklēšanas un birku lasīšanas/rakstīšanas, ar aizbāžņiem izdevās simulēt pilnu plānoto funkcionalitāti tikai sarežģītākā, nepraktiskākā veidā.

Programmatūru plānots turpināt nākamajā ciklā, to veiksmīgi sajūdzot ar DreamFactory WEB servisu, un sniedzot jaunu funkcionalitāti.

Izmantotā literatūra

1. Qt 5 dokumentācija. [tiešsaiste]. [Skatīts 29.05.2016]. Pieejams: http://doc.qt.io/qt-5/
2. QML piemērs kartes integrēšanai. [Skatīts 29.05.2016]. Pieejams: http://doc.qt.io/qt-5/qtlocation-mapviewer-example.html
3. Reading data from TableView with C++ model in it. [Skatīts 29.05.2016]. Pieejams: https://forum.qt.io/topic/67476/reading-data-from-tableview-with-c-model-in-it
4. Recompiling ODBC on Android. [Skatīts 29.05.2016]. Pieejams: https://forum.qt.io/topic/67143/recompiling-odbc-on-android/1
5. DreamFactory tutorials. [Skatīts 29.05.2016]. Pieejams: http://wiki.dreamfactory.com/DreamFactory/Tutorials

Pielikums

NFClues lietotnes ceļvedis

Lai lietotu šo NFClues Jums būs nepieciešams Androīd viedtālrunis ar vismaz 4.4.0 (Kitkat) versiju, kā arī tam jābūt aprīkotam ar NFC uztvērēju/raidītāju. Ja neesat drošs vai Jūsu ierīcei ir NFC, varat pārbaudīt pilno sarakstu ar ierīcēm, kurām NFC ir pieejams šeit: https://en.wikipedia.org/wiki/List_of_NFC-enabled_mobile_devices.

Pievienojoties lietotnei Tev būs nepieciešams reģistrēties kā jaunam lietotājam, lietotājs būs Tava persona NFClues sistēmā, uz tā tiks krāti punkti, izpildīto piedzīvojumu vēsture un paša izveidotie piedzīvojumi.

Galvenais NFClues mērķis ir ļaut lietotajiem veidot pašam savus aizraujošus piedzīvojumus interesantās pasaules vietās, lai izveidotu savu piedzīvojumu Tev būs nepieciešama NFC birka. Tās varēsi iegadāties no ļoti daudz dažādiem piedāvātājiem, mēs iesakām - http://www.whiztags.com/

Kamēr gaidi savu NFC birku vari sākt plānot sava piedzīvojuma tēmu un vietu. Piedzīvojumam jābūt publiski pieejamam, centies to ieplānot interesantā un droši pieejamā vietā, ja esi pieslēdzies, tad caur izvēlni Tev ir iespēja izveidot jaunu piedzīvojumu “Create adventure” sadaļā. Šajā logā Tu vari izpauzties, sniedzot piedzīvojumam nosaukumu, veidojot aprakstu un pavedienu. Lai Tavs piedzīvojums gūtu labākās atsauksmes, piedzīvojuma nosaukumam vajadzētu būt saistītam ar atrašanās vietu, skanīgam un atmiņā paliekošam. Piedzīvojuma aprakstā Tev jāraksturo, kur birka ir paslēpta, paužot to caur mīklām vai abstraktiem stāstiem. Pavediens ir meklētājiem, kuri birku nevar atrast, šeit Tu vari pierakstīt, kādu zīmīgu norādi ar kuru birku būs vieglāk atrast. Apbalvojumā nepieciešams norādīt, cik punktus meklētājs saņems, ja atradīs birku, neesi skops un neesi pārāk dāsns.

Kad šī informācija ir savadīta Tu vari šo piedzīvojumu saglabāt un tas būs pieejams “Your adventures” skatā. Ja Tu uz tā uzspiedīsi, tiksi aizvests uz šī piedzīvojuma apraksta lapu. Tagad, ja esi saņēmis birku, Tev ir iespēja izveidot savu piedzīvojumu. Ar savu Androīd ierīci un tukšo NFC birku dodies uz vietu, kur esi ieplānojis ievietot savu piedzīvojumu. Kad esi tur nonācis, atrodi, kādu sausu, taisnu, tīru virsmu, kurai varēs birku pielīmēt. Kad esi to atradis, atver šo piedzīvojumu lietotnē un spied inicializēt. Ierīce Tev prasīs, lai Tu tai tuvumā pieliec birku. Kad ierīce uztvers birku, lietotne sāks tajā rakstīt unikālu piedzīvojuma atslēgu. Kad tas būs izdarīts, lietotne paziņos, ka piedzīvojums ir inicializēts. Tagad šo birku pielīmē sausajai virsmai, ko izvēlējies iepriekš un ļauj tai gaidīt jaunus meklētājus.

Programmprodukta pirmkoda fragments

```
#ifndef NFCHANDLER_H
#define NFCHANDLER_H

#include <QObject>
#include <QNearFieldManager>
#include <QNdefMessage>
#include <QNdefNfcTextRecord>
#include <QNearFieldShareTarget>
#include <QNearFieldTarget>
#include <QDebug>

/* This class is used for handling all tag related operations
 * writing tag
 * reading tag
 * Handels signals and slots for NFC actions
 */
class NFCHandler : public QObject
{
    Q_OBJECT
public:
    explicit NFCHandler(QObject *parent = 0);
    Q_PROPERTY(int tagId READ tagId WRITE setTagId NOTIFY tagIdChanged)
    Q_PROPERTY(QNdefNfcTextRecord nfcText READ nfcText WRITE setNfcText
NOTIFY nfcTextChanged)
    Q_PROPERTY(QNdefMessage nfcMessage READ nfcMessage WRITE setNfcMessage
NOTIFY nfcMessageChanged)
    Q_PROPERTY(QString errorString READ errorString /*NOTIFY errorChanged*/)

    //Getters
    int     tagId();
    QNdefNfcTextRecord nfcText();
    QNdefMessage nfcMessage();
    QString errorString();

    //Setters
    void    setTagId(const int &tagId);
    void    setNfcText(const QNdefNfcTextRecord &nfctext);
    void    setNfcMessage(const QNdefMessage &nfcMessage);

    //Methods
    Q_INVOKABLE void startReading();
    Q_INVOKABLE void startWriting();
    Q_INVOKABLE void stopLooking();
    Q_INVOKABLE void addText(QNdefNfcTextRecord textRecord);

signals:
    void tagIdChanged();
    void nfcTextChanged();
    void nfcMessageChanged();
    void gotError(QString);
    void error();
```

```
public slots:
    void targetDetected(const QNdefMessage &message, QNearFieldTarget
*target);
    void targetLost(QNearFieldTarget *target);
    void targetError(QNearFieldTarget::Error error, const
QNearFieldTarget::RequestId &id);
protected slots:
    void handleError(QString p_error);
private:
    //Tag data
    int     l_tagId;
    QString l_message;
    QString l_error;
    QNdefMessage                l_nfcMessage;
    QNdefNfcTextRecord          l_nfcText;


    //
    QNearFieldManager           *l_manager;
    QNearFieldTarget            *l_target;
    QNearFieldTarget::RequestId l_request;

    //states
    enum Action {
        NoAction,
        ReadNdef,
        WriteNdef
    };

    Action  l_action;
};
```

```
#endif // NFCHANDLER_H
```

NFCHANDLER_CPP

```
#include "nfchandler.h"
#include <QObject>
#include <QNearFieldManager>
#include <QNdefMessage>
#include <QNdefNfcTextRecord>
#include <QNearFieldShareTarget>
#include <QNearFieldTarget>
#include <QDebug>

NFCHandler::NFCHandler(QObject *parent) : QObject(parent)
{
    connect(this,SIGNAL(gotError(QString)), this,SLOT(handleError(QString)));
}
//Geters////////////

int NFCHandler::tagId()
{
    return l_tagId;
}

QNdefNfcTextRecord NFCHandler::nfcText()
{
    return l_nfcText;
}

QNdefMessage NFCHandler::nfcMessage()
{
    return l_nfcMessage;
}

QString NFCHandler::errorString()
{
    return l_error;
}

//Setters/////////////
void NFCHandler::setTagId(const int &tagId)
{
    if (l_tagId == tagId)
        return;
    l_tagId = tagId;
    emit tagIdChanged();
}

void NFCHandler::setNfcText(const QNdefNfcTextRecord &nfctext)
{
    if (nfctext == l_nfcText)
        return;
    l_nfcText = nfctext;
    emit nfcTextChanged();
}

void NFCHandler::setNfcMessage(const QNdefMessage &nfcMessage)
```

```
{
    if (nfcMessage == l_nfcMessage)
        return;
    l_nfcMessage = nfcMessage;
    emit nfcMessageChanged();
}
//Methods//////////////
/// \brief NFCHandler::startReading
/// Initiated from QMl to tell backend to start looking for tag to read from
void NFCHandler::startReading()
{
    l_action = ReadNdef;
    l_manager->setTargetAccessModes(QNearFieldManager::NdefReadTargetAccess);
    l_manager->startTargetDetection();
}
/// \brief NFCHandler::startWriting
/// Initiated from QMl to tell backend to stop looking for tags to write in
void NFCHandler::startWriting()
{
    l_action = WriteNdef;
    l_manager-
>setTargetAccessModes(QNearFieldManager::NdefWriteTargetAccess);
    l_manager->startTargetDetection();
}
//Slots////////////////
/// \brief NFCHandler::targetDetected
/// Slot called form QNearFieldManager when a tag has been detected
/// \param message - filled with a message if it's returned from manager
/// \param target - target tag that is detected
void NFCHandler::targetDetected(const QNdefMessage &message, QNearFieldTarget
*target)
{
    switch (l_action) {
    case NoAction:
        break;
    case WriteNdef:
        connect(target,
SIGNAL(error(QNearFieldTarget::Error,QNearFieldTarget::RequestId)),
                this,
SLOT(targetError(QNearFieldTarget::Error,QNearFieldTarget::RequestId)));
        connect(target, SIGNAL(ndefMessagesWritten()), this,
SLOT(ndefMessageWritten()));

        l_request = target->writeNdefMessages(QList<QNdefMessage>() <<
l_nfcMessage);
        if (!l_request.isValid()) // cannot write messages
            targetError(QNearFieldTarget::NdefWriteError, l_request);
        break;
    case ReadNdef:
        connect(target,
SIGNAL(error(QNearFieldTarget::Error,QNearFieldTarget::RequestId)),
                this,
SLOT(targetError(QNearFieldTarget::Error,QNearFieldTarget::RequestId)));
        connect(target, SIGNAL(ndefMessageRead(QNdefMessage)),
                this, SLOT(ndefMessageRead(QNdefMessage)));
```

```
        l_request = target->readNdefMessages();
        if (!l_request.isValid()) // cannot read messages
            targetError(QNearFieldTarget::NdefReadError, l_request);
        break;
    }
}
///
/// \brief NFCHandler::targetLost
/// A slot called from QNearFieldManager when a target has been lost
/// \param target - pointer to the lost target
void NFCHandler::targetLost(QNearFieldTarget *target)
{
    target->deleteLater();
}
///
/// \brief NFCHandler::targetError
/// Slot called from targetDetected if an error has occured during NFC
transaction
/// \param error - the type of error
/// \param id - the NFc request that failed
///
void NFCHandler::targetError(QNearFieldTarget::Error error, const
QNearFieldTarget::RequestId &id)
{
    if (l_request == id) {
        switch (error) {
        case QNearFieldTarget::NoError:
            qDebug()<<"No error";
            break;
        case QNearFieldTarget::UnsupportedError:
            gotError("Sorry, Unsupported tag");
            break;
        case QNearFieldTarget::TargetOutOfRangeError:
            gotError("Tag has been removed from the field");
            break;
        case QNearFieldTarget::NoResponseError:
            gotError("No response from the tag");
            break;
        case QNearFieldTarget::ChecksumMismatchError:
            qDebug()<<"Checksum mismatch";
            gotError("Ooops, there seems to be a problem");
            break;
        case QNearFieldTarget::InvalidParametersError:
            qDebug()<<"Invalid parameters";
            gotError("Ooops, there seems to be a problem");
            break;
        case QNearFieldTarget::NdefReadError:
            qDebug()<<"NDEF read error";
            gotError("Ooops, there seems to be a problem");
            break;
        case QNearFieldTarget::NdefWriteError:
            qDebug()<<"NDEF write error";
            gotError("Ooops, there seems to be a problem");
            break;
```

```
        default:
            gotError("Ooops, there seems to be a problem");
        }

        l_manager->setTargetAccessModes(QNearFieldManager::NoTargetAccess);
        l_manager->stopTargetDetection();
        l_request = QNearFieldTarget::RequestId();
    }
}

void NFCHandler::handleError(QString p_error)
{
    qDebug() << "Error happened";
    if (p_error == l_error)
    {
        return;
    }
    else
    {
        l_error = p_error;
        emit error();
    }
}

void NFCHandler::stopLooking()
{
    l_manager->stopTargetDetection();
}
///
/// \brief NFCHandler::addText
/// Used to add text type record to the message being written in a tag
/// \param textRecord - The record that should bee added to the message
///
void NFCHandler::addText(QNdefNfcTextRecord textRecord)
{
    l_nfcMessage.append(l_nfcText);
}
```

Kvalifikācijas darbs **““Near Field Clues”, uz Ģeolokāciju un NFC tehnoloģijām balstīta izpētes spēle”** 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: ***Hermanis Verhoustinskis*** _____ **___.05.2016**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs**: Dr.dat., Guntis Arnicāns** _____ **___.05.2016.**

Recenzents: **M.dat., Jānis Vempers**

Darbs iesniegts 30.05.2016.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2016. prot. Nr. _____

Komisijas sekretārs(-e): _____