LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

Rojas novada tūrisma iOS lietotnes izveide

KVALIFIKĀCIJAS DARBS

Autors:

Armands Baurovskis

Studenta apliecības Nr.:

ab11196

Darba vadītājs:

asociētais profesors, Dr. dat. Uldis Straujums

RĪGA 2013

ANOTĀCIJA

Kvalifikācijas darbā – „Rojas novada tūrisma iOS lietotnes izveide” ir izstrādāta Rojas novada tūrisma lietotne iOS operētājsistēmai – „Roja Travel Guide”. Izstrāde tika veikta ar spējo programmatūras izstrādes metodi. Lietotne nodrošina informāciju divās valodās – latviešu un angļu – par Rojas novada tūrisma objektiem, kā arī ļauj lasīt jaunākās ziņas, kas ir saistītas ar novadu. Lietotne ir paredzēta tūristiem un vietējiem iedzīvotājiem. Lietotne strādā pilnvērtīgi uz visiem *iPhone* un *iPad* modeļiem, kuri atbalsta *iOS 5* versiju. Informācijas apmaiņa tiek veikta, izmantojot Rojas novada *AmberKit* kontu – lietotņu pārvaldības sistēma, kuru izstrādājusi „AmberPhone”.

Atslēgvārdi: *iPhone*, Rojas novads, *iOS* lietotne, tūrisms, spējā programmatūras izstrāde, MVC

ANNOTATION

In the qualification work – „Roja District tourism *iOS* application development" a mobile application – „Roja Travel Guide" about Roja district was developed. The application runs in iOS operating system, it was developed using Agile programming method. The application provides information in two languages – Latvian and English – about Roja district tourist facilities and latest news about the district. The application is designed for tourists and locals. The application works properly on all *iPhone* and *iPad* models that support *iOS 5* version. The exchange of information is done through a Roja district account at AmberKit – the management system for applications, which is developed by "AmberPhone".

Keywords: *iPhone*, Roja District, *iOS* application, tourism, Agile programming method, MVC

SATURS

APZĪMĒJUMU SARAKSTS UN DEFINĪCIJAS

iOS – Informācijas tehnoloģiju uzņēmuma „Apple" izstrādāta operētājsistēma, kas paredzēta „Apple" izgatavotajām mobilajām iekārtām.

Mac OS X – Informācijas tehnoloģiju uzņēmuma „Apple" izstrādāta operētājsistēma, kas paredzēta „Apple" izgatavotajiem personālajiem datoriem un darbstacijām.

UIKit – *iOS* interfeisa struktūra.

Foundation – *Mac OS X* struktūra.

Spējā programmatūras izstrādes metode (Agile) – Informācijas sistēmu ātra izstrāde un ieviešana, ar pakāpeniskiem tuvinājumiem uzlabojot šīs sistēmas un iekļaujot tajās jaunākās lietotāju prasības. Programmas plānošanas gaitā sākotnējai specifikācijai nav jābūt visaptverošai, tajā jāietver tikai tās prasības, kas ir absolūti nepieciešamas. Prasību detalizācija un programmatūras tālāka attīstība tiek veikta tālākos attīstības posmos, tomēr, pirms programmatūra tiek izplatīta, tai jābūt darboties spējīgai un tajā nedrīkst būt kļūdu.

Lietotājstāsts – Uzdevums, ko klients uzdod izstrādātājam spējā programmizstrādes metodē. Parasti tiek uzdots vienkāršā formā, piemēram, „Lietotājs var rediģēt savu profila informāciju".

Iterācija – viens spējās programmatūras metodes izstrādes „cikls" no lietotājstāsta saņemšanas līdz nākamās versijas demonstrēšanai klientiem.

Akcepttests – testa veids, kas ir paredzēts lietotājstāsta pārbaudei pēc tā izstrādes. Tā ir secīga darbību virkne, kas apraksta veicamo uzdevumu darbību un sagaidāmo rezultātu.

Vienībtestēšana – process, kas ietver testēšanas plānošanu, testu kopas iegūšanu un testējamās programmatūras vienības mērīšanu, salīdzinot testējamo vienību ar tai noteiktajām prasībām.

iPhone – Informācijas tehnoloģiju uzņēmuma „Apple" izstrādāts viedtālrunis.

iPad – Informācijas tehnoloģiju uzņēmuma „Apple" izstrādāts planšetdators.

AmberKit – lietotņu pārvaldīšanas sistēma, kuru ir izstrādājusi SIA „AmberPhone"(sīkāk skatīt 2.pielikumā)

SVN(subversion control) – versijas kontroles sistēma, kura glabā projekta veiktās izmaiņas un nepieciešamības gadījumā ir ērti iegūt vecākas revīzijas. Sistēma ļauj ērti sekot veiktajām izmaiņām.

Xcode – „Apple'' izstrādāta programmatūras izstrādes vide, kura ļauj veidot, kompilēt un testēt programmatūru Mac OS X un iOS operētājsistēmām.

Xcode Instruments – Xcode palīgrīks, ar kuru ir iespējams testēt lietotnes veiktspēju, izmantotās atmiņas daudzumu, atmiņas noplūdes gadījumus u.c. problēmas.

Objective–C – objektorientēta programmēšanas valoda, kuru izstrādājusi uzņēmums „Apple", kas galvenokārt tiek izmantota Mac OS X vai iOS operētājsistēmu programmatūru izveidē.

HUD – grafiska lietotnes saskarne, kas uzrāda lietotājam aktuālu informāciju – paziņojumu, brīdinājumu u.c.

UIView – *UIKit* bibliotēkas klase, kura definē taisnstūra veida apgabalu uz ierīces ekrāna, kurā tiek glabāti citi saskarnes elementi.

UIScrollView – Lietotāja saskarnes elements, kas ļauj uzrādīt informāciju, kura ir lielāka nekā ierīces ekrāna izmēri.

UITableView – Lietotāja saskarnes elements, kas ļauj uzrādīt informāciju tabulas veidā.

NSLog() – *Foundation* bibliotēkas funkcija, kura veic teksta izvadi konsoles vidē.

Google Analytics – „Google" izveidota sistēma, kas ļauj iegūt statistiskus datus par lietotnes lietošanu.

Push notifikācija(Push notification) – Mobilo iekārtu ziņu paveids, kurš ir piesaistīts katrai lietotnei. Ziņa tiek saņemta neatkarīgi no tā, vai lietotne ir ieslēgta, vai nē. Pārsvarā tiek izmantota, lai nosūtītu lietotājiem ziņu par lietotnes uzlabojumiem.

Xcode ARC – „Apple'' izstrādāts atmiņas pārvaldības serviss, kas veic lietotnes automātisku atmiņas pārvaldību.

Model – komponente, kas satur klases un saskarnes, kura atbalsta datu modeli.

View – komponente, kas satur klases un saskarnes, kura nodrošina modeļa vizuālo attēlojumu.

Controller – komponente, kas veic pieprasījumu apstrādi no *Model* vai *View*.

IEVADS

Viedtālruņu izmantošana sadzīvē strauji pieaug un mobilo iekārtu lietojumprogrammatūra kļūst arvien populārāka [Ramanathan2012]. „Apple" ražotie viedtālruņi ir kļuvuši ļoti pieprasīti [Singh2013], tāpēc pieprasījums pēc „Apple" viedtālruņu lietotnēm strauji palielinās. Daudzi klienti mēģina uzlabot savu mārketingu, izmantojot mobilo iekārtu lietotnes. Rojas novada dome vēlas popularizēt sava novada tūrismu, izmantojot tieši mobilo iekārtu lietojumprogrammatūru.

Šī kvalifikācijas darba mērķis ir izstrādāt Rojas novada tūrisma *iOS* lietotni, kura popularizēs novada tūrismu.

Rojas novada mājas lapa piedāvā jaunākās ziņas par novadu un detalizētu informāciju par novada tūrisma objektiem. Lietotnes uzdevums ir nodrošināt informāciju divās valodās (latviešu un angļu) par Rojas novada tūrisma objektiem, kā arī ļaut lasīt jaunākās ziņas, kas ir saistītas ar novadu.

Programmēšana notika *Objective – C* valodā, izmantojot *Xcode* vidi. Izstrāde tika veikta, izmantojot spējo programmatūras izstrādes metodiku. Lietotne tika veidota pēc *MVC* arhitektūras modeļa principa. Veidošanas laikā daudz uzmanības tika pievērsts koda kvalitātei, kā arī *Objective – C* kodēšanas standartu ievērošanai.

Dokuments ir sadalīts vairākās nodaļās: funkcionālās prasības, lietotājstāsti, programmatūras projektējuma apraksts, testēšanas dokumentācija. Tiek aprakstīti izstrādes un kvalitātes nodrošināšanas procesi, dažādas darba organizācijas aktivitātes. Pielikumā ir iekļauts programmatūras koda fragments priekšstatam par sistēmas implementācijas izskatu, koda struktūru un komentēšanu, un apraksts par lietotņu pārvaldības sistēmu *AmberKit*, un lietošanas instrukcija.

Darba uzdevumi:

- Veikt analīzi par *iOS* versijas, programmēšanas valodas, arhitektūras modeļa piemērotību lietotnei,
- Izstrādāt lietotājstāstus, vadoties pēc klientu vēlmēm,
- Izstrādāt katras iterācijas plānu un lietotnes prototipu, vadoties pēc iterācijas plāna,
- Veikt akcepttestus pēc katras lietotnes prototipa izveides,
- Testēt lietotnes veiktspēju un labot kļūdas,
- Publicēt lietotni *AppStore*.

1. FUNKCIONĀLĀS PRASĪBAS

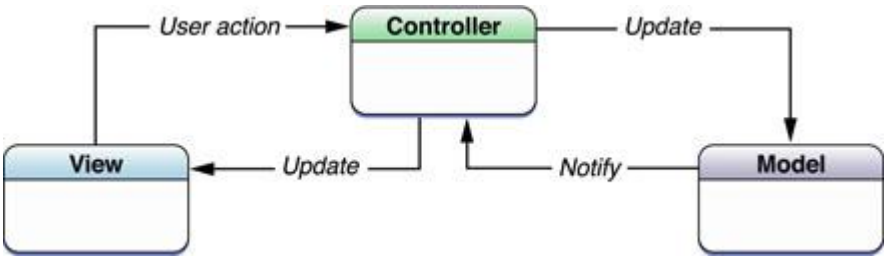
Izstrādājot lietotni „Roja Travel Guide”, ir nepieciešams veikt divus sākuma posmus:

- Izvēlēties programmēšanas valodu;
- Izvēlēties lietotnes zemāko atbalstāmo *iOS* versiju.

Populārākās *iOS* lietotnes izstrādes programmēšanas valodas ir ***C, C++, JavaScript un Objective – C***. Oficiālā *iOS* lietotņu izstrādes vide ir *Xcode*, kuras pamatvaloda ir *Objective – C*. *Xcode* piedāvā plašu ietvaru izmantošanu, kuri balstās uz *Objective – C* programmēšanas valodu, tāpēc šī kvalifikācijas darba ietvaros tika izvēlēta tieši *Objective – C* programmēšanas valoda.

Lai nodrošinātu lietotnes darbību uz *iPhone 5* viedtālruņiem, zemākā pieļaujamā *iOS* versija ir 4.3. Pēc statistikas datiem pasaulē 98% „Apple” mobilās iekārtas izmanto *iOS* 5 un augstāku versiju[Smith2013]. *iOS 5* versija piedāvā ērtāku skatu izveidi un atbalsta *ARC* sistēmu, kas būtiski uzlabo lietotnes veiktspēju[Apple2013a], tāpēc kvalifikācijas darba ietvaros tika izvēlēta *iOS* 5 versija kā zemākā atbalstāmā versija.

Lietotne „Roja Travel Guide” tika veidota pēc *MVC* principa. *MVC(Model–View–Controller)* ir lietotņu arhitektūras struktūra, kura norāda kā būtu nepieciešams nodalīt lietotnes darbības organizētā veidā. *Model* var definēt kā lietotnes „smadzenes” un tas tiek izmantots, lai veiktu dažādas apstrādes darbības kā datu lejupielādēšanu un parsēšanu. *View* ir lietotāja saskarne ar lietotni. Tajā tiek definēti visi skata elementi un *View* reģistrē lietotāja darbības, piemēram, nospiesta poga. *Controller* veic komunikāciju starp *View* un *Model*. Ja lietotājs nospiež pogu, *View* to reģistrē un nosūta informāciju *Controller*. Pēc tam, *Controller* to nosūta *Model*, kas veic datu apstrādi. Kad apstrāde ir pabeigta, *Model* sūta informāciju *Controller*, kas pēc tam atjaunina *View* skata elementus. Lietotnei ir iespējami vairāki šo elementu pāri, tāpēc *Controller* spēj arī komunicēt ar citiem *Controller*.



1.1. att. **MVC darbības principi** [Apple2013c]

2. LIETOTĀJSTĀSTI

Spējās izstrādes metode paredz, ka programmatūras prasības tiek izvirzītas visā programmatūras izstrādes laikā. Katra prasība tiek definēta kā jaunas funkcionalitātes ieviešana vai iepriekš izveidotās funkcionalitātes papildināšana vai labošana. Katram lietotājstāstam ir izveidots akcepttests un tiek novērtēts ar noteiktu sarežģītības novērtējumu.

Lietotājstāsti tiek sadalīti vairākās iterācijās, kuru vidējais ilgums ir līdz divām nedēļām. Katrs lietotājstāsts tiek sadalīts mazākos uzdevumos (blokos) un katram blokam ir izveidots savs akcepttests. Tikai pēc sekmīgi pabeigta uzdevuma un veiksmīgas akcepttestu izpildīšanas var uzsākt jaunu iterācijas ciklu.

Lietotājstāti tiek papildināti visas izstrādes laikā, kurus izstrādātājs novērtē ar sarežģītības punktiem.

Lietotājstāstu autors ir uzņēmuma SIA „AmberPhone" projekta vadītājs Didzis Andersons.

Zemāk ir aplūkojamas iterāciju tabulas ar izveidotajiem lietotājstāstiem.

2.1. tabula

1. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Lietotne tiek optimizēta darbībai iPhone versijai 4. Lietotnes simulācija pārbaudīta līdz iOS versijai 6. Lietotne darbojas, sākot no iOS versijas 5.0.	**2**
Lietotne tiek nodrošināta 2 valodās – latviešu un angļu valodā.	**4**
Valodas maiņa tiek veikta, izmantojot lietotnes iestatījumus.	**3**
Satura nomaiņu lietotnē iespējams veikt caur AmberKit platformas palīdzību, kurā jāievada visi attēlojamie dati.[1]	
Lietotnes dizains ir pielāgots visiem iPhone modeļu ekrāniem: • 320x480 pikseļi; • 640x960 pikseļi; • 640x1136 pikseļi.	**2**

[1] Šis konkrētais lietotājstāsts ir paredzēts PHP programmētājam

Zemāk var aplūkot 2. iterāciju, kura balstās uz kategoriju skatu izveidi.

2.2. tabula

2. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Sākuma skatā tiek attēlotas 8 izvēles iespējas – Ko darīt, Ko redzēt, Kur palikt, Kur paēst, Ziņas, Informācija, Karte, Maršruti.	**4**
Sadaļu – Ko darīt, Ko redzēt, Kur palikt, Kur paēst, Maršruti – pirmais skats ietver objekta: • Attēlu; • Nosaukumu; • Attālumu no lietotāja līdz objektam. Ja ir iespējams, tad ir jāaprēķina attālums no objekta līdz lietotājam. Objekti sakārtoti pēc attāluma augošā secībā.	**7**
Nospiežot uz objekta, par katru vietu tiek uzrādīts detalizēts apraksts un kontaktinformācija: • Nosaukums; • Attēls; • Preformatēts apraksts; • Kontaktinformācija (adrese, tālrunis, mob. tel., e–pasts, mājas lapa).	**7**
Nospiežot uz kontaktinformācijas – telefona numuriem vai mājas lapām – attiecīgi automātiski atveras telefona zvanīšanas funkcija vai tīmekļa pārlūks.	**3**
Lietotājs var atstāt komentāru vai atsauksmi par konkrēto objektu. Ja ir norādīts objekta e–pasts, tad šo ziņu lietotne nosūta uz šo e–pastu, pretējā gadījumā, tā tiek nosūtīta uz kopējo e–pastu.	**3**

Zemāk var aplūkot 3. iterāciju, kura balstās uz jaunumu sadaļu izveidi.

2.3. tabula

3. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Jaunumi un aktualitātes ir jāuzrāda vizuāli baudāmā formā. Datu iegūšanai tiek izmantots AmberKit platformai pieejamais XML, kas satur: • Saiti uz raksta oriģinālo avotu; • Preformātētu aprakstu; • Attēlu; • Publicēšanas datumu.	**5**

Pārskatāmā veidā attēlots pašvaldības *twitter* konts, izvadot pēdējos 10 ierakstus. Tiek norādīts ieraksta publicēšanas datums un teksts.	**5**

Zemāk var aplūkot 4. iterāciju, kura balstās uz kartes un informācijas skatu izveidi.

2.4. tabula

4. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Kartē, kurā pa atbilstošajām kategorijām – Ko darīt, Ko redzēt, Kur palikt, Kur paēst – ir parādītas apmeklējamās vietas, ja pie objekta ir norādītas koordinātes. Iespējams skatīties visas kategorijas vai kādas no tām atspējot.	**7**
Ir jāizveido informācijas lauks, kurā tiek attēlota bilde un pašvaldības iepriekš sagatavots teksts vizuāli baudāmā formā.	**2**

Zemāk var aplūkot 5. iterāciju, kura balstās uz *Google Analtycs* statistikas nodrošināšanu.

2.5. tabula

5. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Lietotnei ir jānodrošina *Google Analytics* statistika.	**2**

Zemāk var aplūkot 6. iterāciju, kura balstās uz izstrādi, kas ļautu lietotnei darboties bezsaistes režīmā.

2.6. tabula

6. Iterācija

Lietotājstāsts	Sarežģītības novērtējums
Lietotnei ir jāstrādā bezsaistes režīmā (visi dati jāglabā uz mobilās ierīces).	**6**

3. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

3.1. Ievads

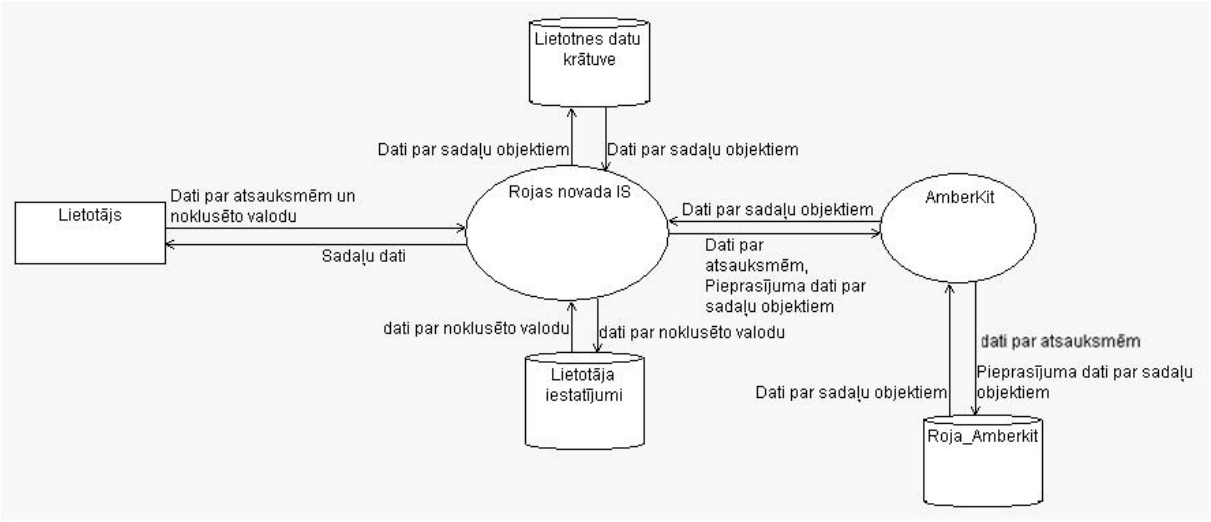
Tā kā programmatūra tika izstrādāta pēc spējās programmatūras metodes, tad netika izstrādāts konkrēts programmatūras projektējuma apraksts projekta sākumā, bet gan izveidots citādāks apraksts, kas spēs sniegt līdzīgu informāciju par lietotni kā tradicionālais PPA.

Tika izveidotas 0. un 1. līmeņa datu plūsmu diagrammas, kurās tiek uzrādītas datu plūsmas starp skatu kontrolieriem un datu krātuvēm. Lai izprastu funkcionalitātes darbību *iOS* vidē, tika izveidots detalizēts funkcionalitātes piemērs, kā arī tika izveidotas klašu hierarhijas un lietotnes skatu diagrammas, un klašu detalizēts apraksts. Klašu detalizēts apraksts ietver klases detalizētu informāciju par tās esošām metodēm un mainīgajiem.

Uzmanība tika vērsta tikai uz autora izstrādātajām klasēm (visas klases nosaukumi, kuri sākas ar identifikatoru „ROJ”).

3.2. 0.līmeņa datu plūsmu diagramma

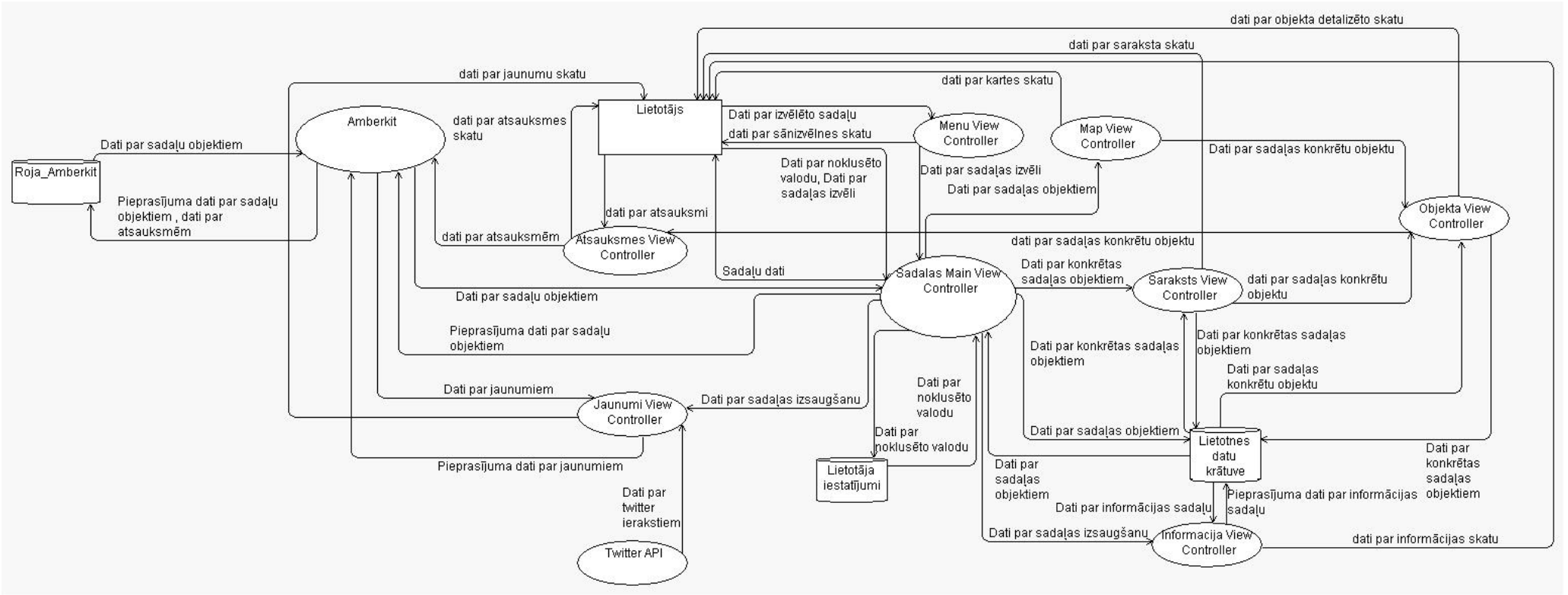
0. līmeņa datu plūsmu diagramma attēlo lietotnes pamatsastāvdaļas. Kopumā sistēma izmanto divas datu krātuves – lietotnes datu krātuve un lietotāju iestatījumu datu krātuvē. Lietotnes datu krātuvē tiek glabāti visi dati par kategorijām – kategoriju detalizēta informācija un attēli. Lietotāju iestatījumu datu krātuvē glabājas noklusētās valodas vērtība un maza izmēra dati, kuri ir specifiski izveidoti katram lietotājam. Lietotnes sistēma arī izmanto ārējo datu krātuvi – *AmberKit,* kurā tiek glabāti un mainīti kategorijas dati. Sistēma satur tikai vienu lietotāju grupu – „Lietotājs".



3.2.1. att. **0.līmeņa datu plūsmu diagramma**

3.3. 1. līmeņa datu plūsmu diagramma

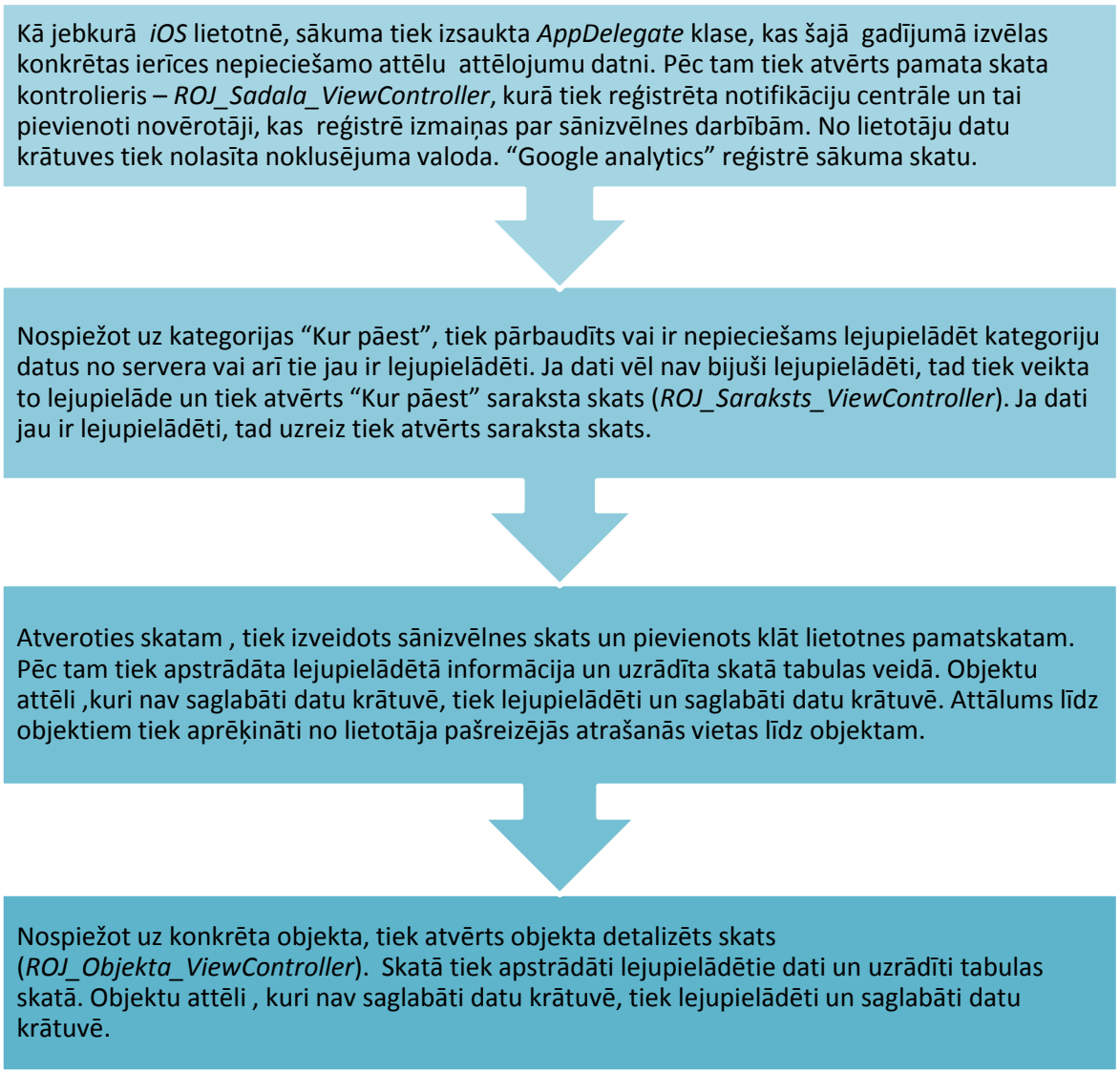
1. līmeņa datu plūsmu diagramma jau plašāk attēlo sistēmas uzbūvi. Šī diagramma sniedz informāciju par datu plūsmām starp skatu kontrolieriem, modeļiem un datu krātuvēm. Lietotājam ir saskarne ar visiem skata kontrolieriem.



3.3.1. att. **1.līmeņa datu plūsmu diagramma**

3.4. Funkcionalitātes piemērs

Lai izprastu lietotnes darbību, tika izveidots detalizēts, grafisks funkcionalitātes piemērs par konkrētu lietotāja darbību. Tiek apskatīts process, kad lietotājs vēlas apskatīt „Kur pāest" kategorijas konkrēta objekta detalizētu informāciju. Tādā gadījumā tiek izsauktas vairākas metodes un skatu kontrolieri. *4.4.1. attēlā* ir redzama šīs funkcionalitātes gaita un darbības.

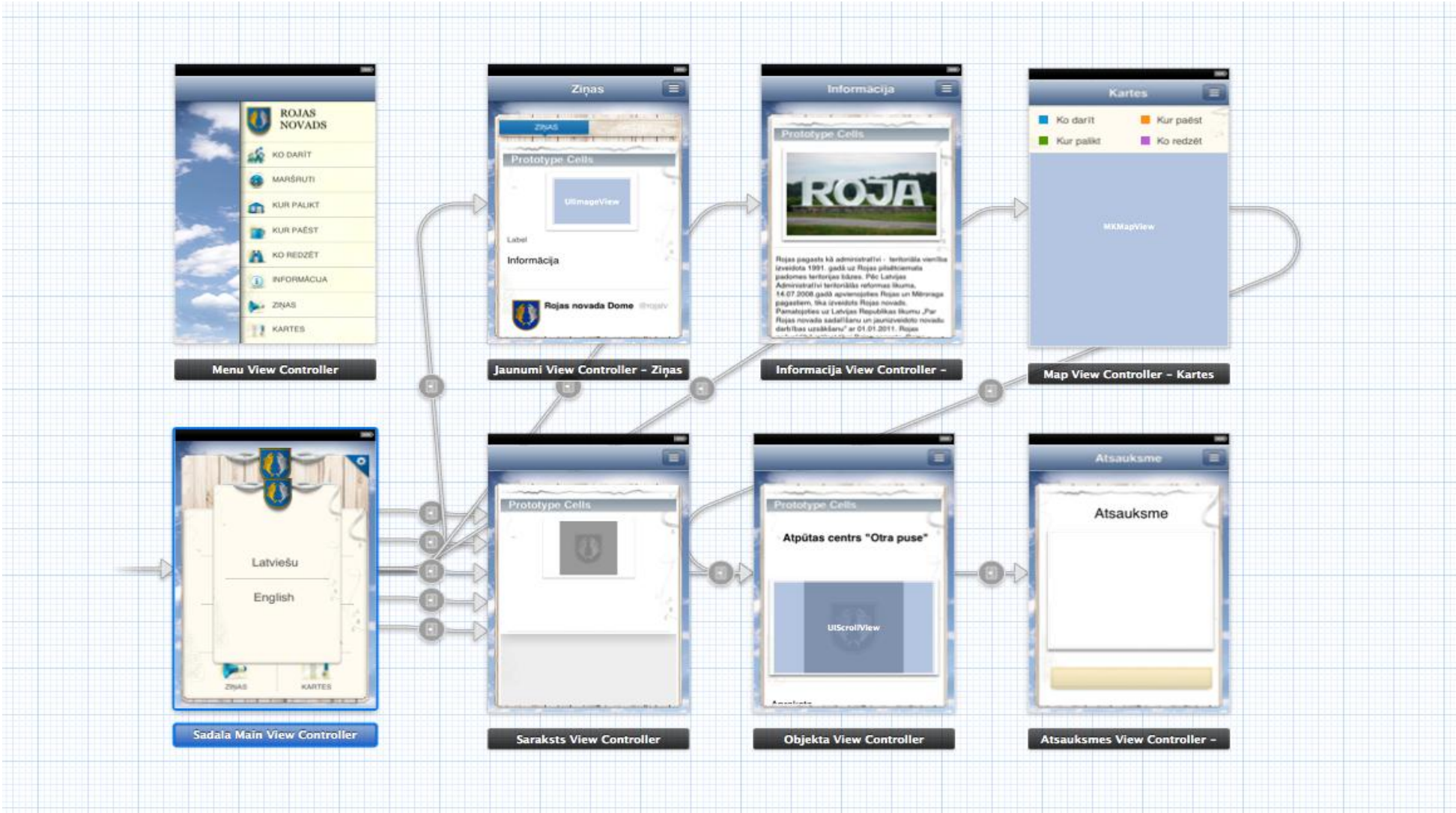


3.4.1 att. **Detalizēta funkcionalitātes gaita un darbības**

3.5. Lietotnes „Roja Travel Guide” skatu un klašu mantošanas diagrammas

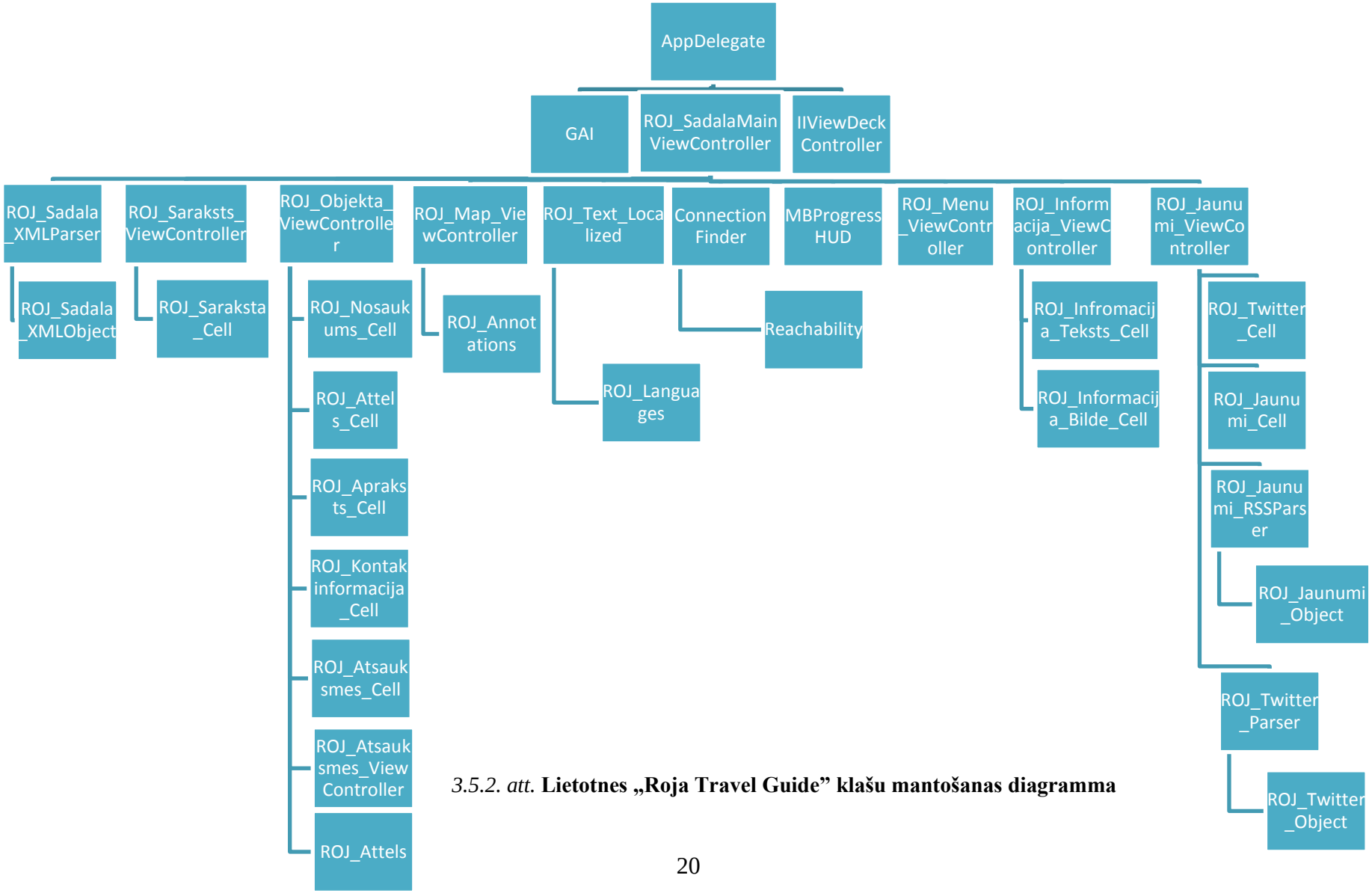
Lietotnes vajadzībām tika izveidoti astoņi skati gan *iPhone 4*, gan *iPhone 5* iekārtām. Skatu diagramma uzskatāmi attēlo lietotnes darbības gaitu un savstarpējo skatu komunicēšanu. Katram skatam tika izveidots savs skata kontrolieris. Kontrolieri ir aprakstīti nākamajā nodaļā.

Visas klases, kuras sākas ar identifikatoru „ROJ”, ir autora veidotas šī projekta vajadzībām. Klašu mantošanas diagrammu var aplūkot 4.5.2. attēlā. Veidojot jaunu projektu, automātiski tika iekļautas vairākas „Apple” izstrādātas struktūras, kuras netika uzrādītas klašu mantošanas diagrammā. Detalizētus klašu aprakstus var aplūkot nākamajā nodaļā.



*3.5.1.att.***Lietotnes „Roja Travel Guide" skatu diagramma**

Zemāk var aplūkot lietotnes „Roja Travel Guide” klašu mantošanas diagrammu, kas detalizēti attēlo klašu hierarhiju.



3.5.2. att. **Lietotnes „Roja Travel Guide” klašu mantošanas diagramma**

3.6. Klašu detalizēts apraksts

Šajā apakšnodaļā tiek detalizēti aprakstītas autora izveidotās klases, klases metodes un klases mainīgie. Daudzas klases izmanto mantotās klases metodes, kuras izstrādāja „Apple", bet tās netika aprakstītas šajā darbā, jo to aprakstus ir iespējams izlasīt *iOS* dokumentācijā. Skata tipa mainīgie (visi mainīgie, kas sākas ar identifikatoru „UI") arī netika aprakstīti.

Galvenais izstrādes palīgmateriāls bija „Apple" *iOS* dokumentācija, kurā ir sīki aprakstītas visas klases un to metodes. Kā papildmateriāli tika izmantotas vairākas grāmatas un interneta vietnes, kuras sniedza plašāku informāciju par vairāku algoritmu izstrādi un dažādu metožu pielietošanu. [Apple2013b], [Mark2011], [Sadun2012], [thenewboston.org2013].

Programmatūrā tika iekļautas trīs klases, kuras nav autora veidotas:

- MBProgressHUD [Bukovinski2013],
- IIViewDeckControler [Adriaenssen2013],
- ConnectionFinder [Million2013].

3.6.1. ROJ_SadalaMain_ViewController

ROJ_SadalaMain_ViewController klase ir lietotnes pamatskata kontrolieris, kurā ir iespējams izvēlēties jebkuru no lietotnes kategorijas skatiem. Šajā skatā ir iespējams mainīt valodu.

Metožu dokumentācija

- –(void)changeLabelTexts

  Šī metode veic skata elementu tekstu maiņa pēc valodas maiņas.

- –(IBAction)getLanguage:(id)sender

  Šī metode tiek izsaukta, kad lietotājs veic valodas maiņu. Tiek lejupielādēts konkrētās valodas XML no servera, un izvēlētā valoda tiek saglabāta kā noklusējuma valoda.

Parametri:

- (id)sender – nospiestās pogas mainīgais.

- –(IBAction)hideView:(id)sender

  Metode, kas paslēpj valodas izvēlnes skatu pēc lietotāja pieprasījumu, izsaucot metodi *hideLanguangeView*.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

- –(void)messageFromMenu

  Šī ir notifikācijas metode, kura tiek izsaukta, kad tiek veikta kategorijas maiņa no *ROJ_Menu_ViewController*. Metode saņem kategorijas parametru un uzrāda tās skatu.

- –(void)hideLanguangeView

  Metode, kura noslēpj valodas izvēles skatu.

- –(NSMutableArray)readDataWithName:(NSString*)name

  Metode, kura iegūst datus no ierīces datu glabātuves, kad tas ir nepieciešams.

  Parametri:

  - (NSString*)name – datnes nosaukums.

- –(BOOL)systemVersion:(NSString*)version

  Metode, kura veic pārbaudi, vai padotās versijas numurs atbilst ar ierīces sistēmas versiju.

  Parametri:

  - (NSString*)version – Sistēmas versija, kuru vēlās pārbaudīt.

- –(void)AddData:(NSMutableArray*)object withName:(NSString*)name

  Metode, kura saglabā datus ierīces datu glabātuvē, kad tas ir nepieciešams.

  Parametri:

  - (NSMutableArray*)object – masīvs ar datiem, kurus nepieciešams saglabāt,
  - (NSString*)name – datnes nosaukums.

- –(void)getData

  Metode, kura lejupielādē informācijas datus no servera un veic parsēšanas metodes izsaukumu. Šī metode strādā citā programmas pavedienā, lai neietekmētu pārējo lietotnes darbību.

- –(void)receiveData

  Metode tiek izsaukta, kad tiek veikta valodas maiņa vai lietotājs pirmo reizi veic kategorijas izvēlni. Pēc valodas maiņas tiek lejupielādēti jaunie dati (tiek izsaukta *getData()* metode).

- –(void)pushView

  Šī metode kontrolē konkrētā kategorijas skata uzrādīšanu.

- –(IBAction)setLanguange:(id)sender

  Metode, kas uzrāda valodas izvēlnes skatu.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

- –(void)getError

  Metode, kas veic vizuālu kļūdu paziņošanu lietotājam.

- –(IBAction)goToNews:(id)sender

  Metode, kura uzrāda „Jaunumu” sadaļas skatu.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

- –(IBAction)pressedOption:(id)sender

  Metode, kas veic datu lejupielādi, ja gadījumā tā nav vēl tikusi veikta un izsauc *pushView* metodi, kas uzrāda lietotāja izvēlētās kategorijas skatu. Šī metode tiek izsaukta, kad lietotājs ir izvēlējies kategoriju.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

Mainīgo dokumentācija

- ROJ_Sadala_XMLParser* infoFromServer – Lejupielādēto datu parsētājs,
- CLLocationManager* locationManager – „Apple" izstrādāts struktūras mainīgais, kurš veic lietotāja atrašanās vietas apstrādi,
- MBProgressHUD* HUD – mainīgas, kas veic vizuālu paziņojumu uzrādīšanu lietotājam skatā,
- NSUserDefaults* defaults – „Apple" izstrādāts struktūras mainīgais, ar kura palīdzību ir iespējams saglabāt datus lietotāju datu glabātuvē,
- int currentLanguage – noklusētās valodas skaitliskais identifikators,
- ROJ_Text_Localized* textLocalization – mainīgais, kas nolasa konkrētās valodas datnē saglabātos vārdus,
- ConnectionFinder* connection – mainīgais, kas veic interneta savienojuma pārbaudi.
- NSString* language – noklusētā valoda,
- int buttonID – lietotāja nospiestās pogas identifikators.

3.6.2. ROJ_Sadala_XMLParser

ROJ_Sadala_XMLParser klase veic lejupielādēto tūrismu objektu datu parsēšanu. Tā izmanto „Apple" izstrādāto standarta *NSXMLParser* bibliotēku ar tās esošajām metodēm. Pēc parsēšanas tiek izveidoti vairāki *ROJ_Sadala_XMLObject* un glabāti masīvā.

3.6.2.1. Tūrismu objektu XML struktūra

Zemāk var aplūkot *XML* struktūru tūrisma objektu ielādei (latviešu un angļu valodām).

```
<objekti>
      <objekts>
            <nosaukums>...<nosaukums>
            <tips>...</tips>
            <apraksts>...</apraksts>
            <bildes>
                  <bilde>...</bilde>
            </bildes>
            <adrese>...</adrese>
            <telefons>...</telefons>
            <mobilaistelefons>...</mobilaistelefons>
            <epasts>...</epasts>
            <majaslapa>...</majaslapa>
            <latitude>...</latitude>
            <longitude>...</longitude>
      </objekts>
      ...
</objekti>
```

Mainīgo dokumentācija

- objekti – tags, kas ietver sevī visus objektus,
- objekts – tags, kas sevī ietver objekta informāciju,
- nosaukums – tags, kas ietver objekta nosaukumu,
- tips – tags, kas ietver tipa nosaukumu (stay, do u.c.),
- apraksts – tags, kas ietver objekta aprakstu,
- bildes – tags, kas ietver objekta bildes,
- bilde – tags, kas ietver bildes URL,
- adrese – tags, kas ietver objekta atrašanās vietas adresi,
- telefons – tags, kas ietver objekta tālruņa numuru,
- mobilaistelefons – tags, kas ietver objekta mobilā tālruņa numuru,
- epasts – tags, kas ietver objekta e–pasta adresi,
- majaslapa – tags, kas ietver objekta mājas lapas URL,
- latitude – tags, kas ietver objekta koordinašu ģeogrāfisko platumu,
- longitude – tags, kas ietver objekta koordinašu ģeogrāfisko garumu.

3.6.3. ROJ_Sadala_XMLObject

ROJ_Sadala_XMLObject klase glabā datus par kategorijas objektiem.

Mainīgo dokumentācija

- NSString* nosaukums – objekta nosaukums,
- NSString* tips – kategorijas nosaukums,
- NSString* apraksts – objekta detalizēts apraksts,
- NSArray* bildes – objektam piesaistītas bildes,
- NSString* telefons – objekta kontaktinformācija: tālrunis,
- NSString* mobilaistelefons – objekta kontaktinformācija: mobilais tālrunis,
- NSString*epasts – objekta kontaktinformācija: e–pasts,
- NSString*majasalapa – objekta kontaktinformācija: mājas lapa,
- NSString*adrese – objekta kontaktinformācija: atrašanās vietas adrese,
- NSNumber*latitude – objekta kontaktinformācija: koordinašu ģeogrāfiskais platums,
- NSNumber*longitude – objekta kontaktinformācija: koordinašu ģeogrāfiskais garums,
- NSNumber*attalums – objekta attālums no lietotāja atrašanās vietas.

3.6.4. ROJ_Saraksts_ViewController

ROJ_Saraksts_ViewController klase ir konkrētas kategorijas objektu skata kontrolieris. Tā attēlo tabulas veidā visus objektus, kas pieder konkrētai kategorijai, sakārtotus augošā secībā pēc attāluma no lietotāja atrašanās vietas. No šī skata pēc tam ir iespējams atvērt konkrēta objekta detalizētu skatu. Lielāko daļu metodes sastāda „Apple" izstrādātās UITableView bibliotēkas metodes, kuras netika aprakstītas.

Metožu dokumentācija

- –(void)readDictionary

  Metode, kura veic *readData()* metodes izsaukumu un veic datu iegūšanu no datu glabātuves.

- –(NSMutableDictionary)readDataWithName:(NSString*)name

  Metode, kura iegūst datus no ierīces datu glabātuves, kad tas ir nepieciešams.

  Parametri:

  - (NSString*)name – datnes nosaukums.

- –(void)AddData:(NSMutableDictionary*)object withName:(NSString*)name

  Metode, kura saglabā datus ierīces datu glabātuvē, kad tas ir nepieciešams.

  Parametri:

  - (NSMutableDictionary*)object – vārdnīca ar datiem, kurus nepieciešams saglabāt.
  - (NSString*)name – datnes nosaukums.

- –(void)getDistance

  Šī metode aprēķina attālumu kilometros starp objektu un lietotāju.

- –(UIImage*)imageForCell:(NSIndexPath*)indexPath object:(ROJ_Sadala_XMLObject)object

  Metode, kas veic datu apstrādi par konkrēta objekta attēlu. Ir iespējams trīs situācijas:

  1. attēls nav ne saglabāts, ne atrodas kešatmiņā.

     Darbības plāns: veic attēla lejupielādi, saglabā ierīces datu glabātuvē un kešatmiņā;

  2. attēls ir saglabāts, bet neatrodas kešatmiņā.

     Darbības plāns: saglabā attēlu kešatmiņā;

  3. attēls ir saglabāts un atrodas kešatmiņā.

     Darbības plāns: iegūst attēlu no kešatmiņas;

  Parametri:

  - (NSIndexPath*)indexPath – objekta atrašanās vieta tabulā,
  - (ROJ_Sadala_XMLObject)object – objekts, kura attēls tiek iegūts.

Mainīgo dokumentācija

- NSMutableArray* dataFromSadala – masīvs ar konkrētas kategorijas objektiem,
- NSString* tips – kategorijas nosaukums,
- ROJ_Sadala_XMLObject* objectForSpecificview – konkrēts kategorijas objekts,
- NSMutableDictionary* photos – vārdnīca ar objektu attēliem,
- NSCache* imageCache – kešatmiņas apgabals, kas paredzēts attēliem,
- CLLocationManager* locationManager – „Apple" izstrādāts struktūras mainīgais, kurš veic lietotāja atrašanās vietu apstrādi,
- MBProgressHUD* HUD – mainīgas, kas veic vizuālu paziņojumu uzrādīšanu lietotājam skatā,
- BOOL canGetGPSValues – mainīgais, kurš definē to vai ir iespējams iegūt lietotāja atrašanās vietas koordinātes,
- ROJ_Text_Localized* textLocalization – mainīgais, kas nolasa konkrētās valodas datnē saglabātos vārdus.

3.6.5. ROJ_Saraksta_Cell

ROJ_Saraksta_Cell klase ir tabulas skata rindas elements. Pārsvarā *_Cell* tipa klasēs tiek definēti tikai skata elementi. Arī šī projekta ietvaros, visas *_Cell* klases satur tikai skata elementus.

3.6.6. ROJ_Objekta_ViewController

ROJ_Objekta_ViewController klase definē konkrēta apskates objekta skatu kontrolieri. Klase ietver tabulas skatu, kas sevī ietver daudzus rindas elementus.

Metožu dokumentācija

- –(CGSize)getHeight:(NSString*)text size:(float)size

  Metode, kura aprēķina teksta ietvara augstumu konkrēta teksta attēlošanai.

  Parametri:

  - (NSString*)name – teksts, kuru nepieciešams attēlot,
  - (float)size – teksta izmērs.

- –(float)getKontaktinformacijaCellSize

  Metode, kura aprēķina tabulas skata kontaktinformācijas rindas elementa nepieciešamo augstumu.

- –(void)downloadPhoto: (ROJ_Attels*)object

  Metode, kura nepieciešamības gadījumā lejupielādē objekta attēlu, ja tas nav saglabāts ierīces datu glabātuvē. Pēc veiksmīgas attēla lejupielādēšanas, metode saglabā attēlu ierīces datu glabātuvē un pievieno to *ScrolView* klases mainīgajam.

  Parametri:

  - (ROJ_Attels*)object – objekts, kurš satur attēla URL. Šis objekts tiek izveidots gadījumā, ja attēls netika atrasts ierīces datu glabātuvē.

- –(void)getPhotosForCell:(ROJ_Attels_Cell*)cell

  Metode, kura iegūst visus objekta attēlus no ierīces datu glabātuves, kuri ir saglabāti. Pretējā gadījuma, tiek izveidots objekts *(ROJ_Attels*)object* un citā pavedienā tiek izsaukta metode *downloadPhoto: (ROJ_Attels*)object*, kura veic attēla lejupielādēšanu.

Parametri:

- (ROJ_Attels_Cell*)cell – Tabulas rindas elements, kurš satur *ScrollView* klases mainīgo ar objekta attēliem.

- –(void)addPhotosToScrollView:(UIScrollView*)scrollview photo:(UIImage*)image

  Metode, kura pievieno attēlu *ScrollView* klases mainīgajam.

  Parametri:

  - (UIScrollView*)scrollview – „Apple" izstrādāta klase, kura ļauj ērti pārslēgties no vienas bildes uz otru, izmantojot *Swipe* komandu.
  - (UIImage*)image – attēla mainīgais.

- –(void)getObjectInfo: (ROJ_Attels_Cell*)cell

  Metode, kura pievieno *ScrollView* klases mainīgo tabulas rindas elementam kā atribūtu.

  Parametri:

  - (ROJ_Attels_Cell*)cell – Tabulas rindas elements, kurš satur *ScrollView* klases mainīgo ar objekta attēliem.

- –(void)reloadData

  Metode, kura atjaunina tabulas skatu.

- –(NSMutableDictionary)readDataWithName:(NSString*)name

  Metode, kura iegūst datus no ierīces datu glabātuves, kad tas ir nepieciešams.

  Parametri:

  - (NSString*)name – datnes nosaukums.

- –(void)AddData:(NSMutableDictionary*)object withName:(NSString*)name

  Metode, kura saglabā datus ierīces datu glabātuvē, kad tas ir nepieciešams.

  Parametri:

  - (NSMutableDictionary*)object – vārdnīca ar datiem, kurus nepieciešams saglabāt,
  - (NSString*)name – datnes nosaukums.

- –(void)sortInfoAtCell:(ROJ_Kontaktinformacija_Cell*)cell

  Metode, kura izmaina kontaktinformācijas atribūtu ietvarus tā, lai iztrūkstošas informācijas gadījumā neveidojas lieki tukšumi tabulas rindas elementā.

  Parametri:

  - (ROJ_Kontaktinformacija_Cell*)cell – Tabulas rindas elements, kurš satur kontaktinformācijas atribūtus (tālruni, adresi, mobilo tālruni, e–pasta adresi, mājas lapu).

- –(IBAction)forward:(id)sender

  Metode, kura veic nākamā attēla uzrādīšanu *ScrollView* skatā.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais

- –(IBAction)back:(id)sender

  Metode, kura veic iepriekšējā attēla uzrādīšanu *ScrollView* skatā.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

- –(UIImage*)readImageWithName:(NSString*)name

  Metode, kura iegūst attēlu no ierīces datu glabātuves.

  Parametri:

  - (NSString*)name – datnes nosaukums

- –(void)AddImageWithName:(NSString*)name image:(UIImage*)image

  Metode, kura saglabā attēlu ierīces datu glabātuvē, kad tas ir nepieciešams.

  Parametri:

  - (UIImage*)image – attēls, kuru nepieciešams saglabāt,
  - (NSString*)name – datnes nosaukums.

Mainīgo dokumentācija

- ROJ_Sadala_XMLObject* object – konkrētais skata objekts,
- int scrollPagePos – *ScrollView* redzamā attēla identifikators,
- CGSize descriptionSize – apraksta ietvara augstums,
- CGSize titleSize – virsraksta ietvara augstums,
- NSMutatbleArray* savedPhotos – masīvs ar attēliem, kuri ir saglabāti ierīces datu glabātuvē,
- NSMutatbleArray* savedNewPhotos – masīvs ar attēliem, kuri ir lejupielādēti, bet nav vēl saglabāti ierīces datu glabātuvē,
- int tagCount – attēlu skaits *ScrollView* skatā,
- int currentLanguage – noklusētās valodas skaitliskais identifikators,
- ROJ_Text_Localized* textLocalization – mainīgais, kas nolasa konkrētās valodas datnē saglabātos vārdus,
- ROJ_Kontaktinformacija_Cell* contactCell – kontaktinformācijas tabulas rindas elements.

3.6.7. ROJ_Nosaukums_Cell, Roj_Attels_Cell, ROJ_Apraksts_Cell, ROJ_Kontaktinfromacija_Cell, ROJ_Atsauksmes_Cell

Visas šīs klases ir *ROJ_Objekta_ViewController* tabulas skata rindas elementi, kuri satur teksta un attēla atribūtus.

3.6.8. ROJ_Attels

ROJ_Attels klase satur tabulas skata rindas elementu un objekta attēlu.

3.6.9. ROJ_Atsauksmes_ViewController

ROJ_Atsauksmes_ViewController ir atsauksmju skata kontroliera klase. Skatā ir iespējams nosūtīt lietotāja rakstītu atsauksmi kādam konkrētam objektam. Ja objekts nesatur e–pasta adresi, tad atsauksme tiek nosūtīta uz kopīgo e–pastu.

Metožu dokumentācija

- –(void)sendFeedbackToServer

  Metode, kura veic POST pieprasījumu serverim ar lietotāja ievadītajiem datiem. Ja atsauksme ir nosūtīta veiksmīgi, tad no servera tiek saņemts apstiprinājums, pretējā gadījumā tiek saņemts kļūmes ziņojums. Abos gadījumos lietotājam tiek paziņots – vai atsauksme ir veiksmīgi nosūta, vai radusies kļūme.

- –(void)getError

  Metode, kas veic vizuālu kļūdu paziņošanu lietotājam.

- –(void)simple:(BOOL)stop succeded:(BOOL)success

  Metode, kura uzrāda lietotājam paziņojumu par atsauksmes nosūtīšanas stāvokli.

  Parametri:

  - (BOOL)stop – Parametrs, kurš nosaka – vai paziņojums ir jāuzrāda, vai jānoņem,
  - (BOOL)success – paramteris, kurš atgriež YES vai NO par atsauksmes veiksmīgu nosūtīšanu.

Mainīgo dokumentācija

- int currentLanguage – noklusētās valodas skaitliskais identifikators.
- ROJ_Sadala_XMLObject* object – konkrētais objekts,
- ROJ_Text_Localized* textLocalization – mainīgais, kas nolasa konkrētās valodas datnē saglabātos vārdus,
- ConnectionFinder* connection – mainīgais, kas veic interneta savienojuma pārbaudi,
- MBProgressHUD* HUD – mainīgas, kas veic vizuālu paziņojumu uzrādīšanu lietotājam skatā.

3.6.10. ROJ_Map_ViewController

ROJ_Map_ViewController ir kartes skata kontroliera klase. Šajā skatā tiek uzrādīti visi objekti kartē un lietotāja atrašanās vieta kartē. Lietotājam ir iespēja izvēlēties, kuras kategorijas objektus rādīt kartē. Izvēloties konkrētu objektu, ir iespējams aplūkot objekta detalizētu informāciju.

Metožu dokumentācija

- –(void)drawPinsOnMap

  Metode, kura veic visu objektu attēlošanu kartē.

- –(void)ZoomLocation

  Metode, kura pietuvina kartes skatu tā, lai lietotājam būtu redzami visi objekti.

- –(void)removePinsInCategory:(int)tag

  Metode, kura noņem visus konkrētās kategorijas objektus no kartes.

  Parametri:

  - (int)tag – kategorijas identifikators.

- –(void)addPinsInCategory:(int)tag

  Metode, kura pievieno visus konkrētās kategorijas objektus kartē.

  Parametri:

  - (int)tag – kategorijas identifikators.

Mainīgo dokumentācija

- NSarray* coordinates – masīvs ar objektiem, kuriem ir norādītas koordinētes.

3.6.11. ROJ_Annotations

ROJ_Annotations ir kartes skata papildklase. Tā tika izveidota, lai varētu identificēt katru kartes objektu. Pēc noklusējuma, kartes objektiem nevar piešķirt identifikatoru un tādēļ tika izveidota apakšklase, kurā tika izveidots papildus objekta mainīgais – identifikators.

3.6.12. ROJ_Text_Localized

ROJ_Text_Localized klase veic datu iegūšanu no lokalizācijas datnēm. Tā kā lietotne ir jānodrošina divās valodās, tika izveidotas lokalizācijas datnes, kurās tika glabāti tulkojumi un to identifikatori. Šī klase veic konkrētas valodas lokalizācijas datnes izvēli un veic vārdu pieprasījumu pēc konkrēta identifikatora.

Metožu dokumentācija

- –(NSString*)keyForLanguange:(NSString*)key

  Metode, kura pēc konkrēta identifikatora atgriež *string* tipa mainīgo no lokalizācijas datnes.

3.6.13. ROJ_Languages

ROJ_Languages klase glabā valodas identifikatorus. Šī klase tika izveidota, lai ērtāk veiktu valodas identificēšanu klasē *ROJ_Text_Localized.*

3.6.14. ROJ_Menu_ViewController

ROJ_Menu_ViewController ir sānizvēles skata kontroliera klase. Klases mērķis: no jebkura lietotnes skata ir iespēja pārslēgties uz citu skatu. Klase sastāv no vienas metodes, kura pārslēdz lietotāja izvēlēto skatu.

Metožu dokumentācija

- –(IBAction)doTask:(id)sender

  Metode, kas uzrāda lietotāja izvēlētās kategorijas skatu.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

3.6.15. ROJ_Informacija_ViewController

ROJ_Informacija_ViewControler ir informācijas sadaļas skata kontrolieris. Skats sastāv no tabulas skata, kurā tiek uzrādīts iepriekš sagatavots teksts par Rojas novadu.

3.6.16. ROJ_Informacija_Teksts_Cell, ROJ_Informacija_Bilde_Cell

Šīs abas klases ir informācijas sadaļas tabulas skata rindas elementi, kuri sastāv attiecīgi no attēla un teksta.

3.6.17. ROJ_Jaunumi_ViewController

ROJ_Jaunuma_ViewController ir jaunumu un *twitter* skata kontroliera klase. Šajā skatā ir iespējams apskatīt Rojas novada jaunumus vai 10 jaunākās Rojas novada *twitter* konta ziņas, pārslēdzot konkrēto skatu.

Metožu dokumentācija

- –(void)getJaunumi

  Metode, kura lejupielādē jaunumu datus no servera un veic parsēšanas metodes izsaukumu. Šī metode strādā citā programmas pavedienā, lai neietekmētu pārējo lietotnes darbību.

- –(void)getTweets

  Metode, kura lejupielādē *twitter* datus no *twitter* servera un veic parsēšanas metodes izsaukumu. Šī metode strādā citā programmas pavedienā, lai neietekmētu pārējo lietotnes darbību.

- –(void)reloadTable

  Metode, kura atjauno tabulas skata datus.

- –(NSMutableArray)getHeightForTweets

  Metode, kura aprēķina katra *twitter* ziņas teksta ietvara augstumu konkrēta teksta attēlošanai un saglabā tos masīvā..

- –(IBAction)pressedButton:(id)sender

  Metode, kura pārslēdz jaunumu vai *twitter* ziņu tabulas skatus.

  Parametri:

  - (id)sender – nospiestās pogas mainīgais.

- –(void)simple:(BOOL)stop

  Metode, kura uzrāda lietotājam aktivitātes indikatoru.

  Parametri:

  - (BOOL)stop – Parametrs, kurš nosaka – vai indikators ir jāuzrāda, vai jānoņem.

Mainīgo dokumentācija

- NSarray* heightZinas – masīvs ar jaunumu ziņu teksta ietvaru augstumiem,
- NSarray* heightTwitter – masīvs ar *twitter* ziņu teksta ietvaru augstumiem,
- ROJ_Jaunumi_RSSParser* parserZinas – lejupielādēto jaunumu datu parsētājs,
- ROJ_Twitter_Parser* parserTweet – lejupielādēto *twitter* ziņu datu parsētājs,
- int currentLanguage – noklusētās valodas skaitliskais identifikators.
- ROJ_Text_Localized* textLocalization – mainīgais, kas nolasa konkrētās valodas datnē saglabātos vārdus,

- MBProgressHUD* HUD – mainīgais, kas veic vizuālu paziņojumu uzrādīšanu lietotājam skatā,
- int currentState – tabulas skata identifikators. Ja identifikatora vērtība ir 0, tad ir atvērts jaunumu skats, ja vērtība ir 1, tad atvērts *twitter* skats.

3.6.18. ROJ_Jaunumi_RSSParser

ROJ_Jaunumi_RSSParser klase veic lejupielādēto jaunumu datu parsēšanu. Tā izmanto „Apple" izstrādāto standarta *NSXMLParser* bibliotēku ar tās esošajām metodēm. Pēc parsēšanas, tiek izveidoti vairāki *ROJ_Jaunumi_Object* objekti un glabāti masīvā.

3.6.18.1. Jaunumu XML struktūra

Zemāk var aplūkot *XML* struktūru jaunumu ielādei.

```
<jaunumi>
      <jaunums>
            <nosaukums>...<nosaukums>
            <apraksts>...</apraksts>
            <bilde>...</bilde>
            <links>...</links>
            <datums>...</datums>
      </jaunums>
      ...
</jaunumi>
```

Mainīgo dokumentācija

- jaunumi – tags, kas ietver sevī visus jaunumus,
- jaunums – tags, kas sevī ietver jaunuma informāciju,
- nosaukums – tags, kas ietver jaunuma virsrakstu,
- apraksts – tags, kas ietver jaunuma tekstu,
- bilde – tags, kas ietver bildes URL,
- links – tags, kas ietver URL uz jaunumu rakstu Rojas novada mājas lapā,
- datums – tags, kas ietver jaunuma publicēšanas datumu(*Unix timestamp*).

3.6.19. ROJ_Jaunumi_Object

ROJ_Jaunumi_Object klase glabā datus par jaunumiem.

Mainīgo dokumentācija

- NSString* title – ziņas virsraksts.
- NSString* link – ziņas oriģinālraksta tīmekļa adrese.
- NSString* author – ziņas autors,
- UIImage* photo – ziņas attēls,
- NSString* description – ziņas teksts,
- NSDate* pubDate – ziņas publicēšanas datums.

3.6.20. ROJ_Twitter_Parser

ROJ_Twitter_Parser klase veic lejupielādēto *twitter* datu parsēšanu. Tā izmanto „Apple" izstrādāto standarta *NSXMLParser* bibliotēku ar tās esošajām metodēm. Pēc parsēšanas, tiek izveidoti vairāki *ROJ_Twitter_Object* objekti un glabāti masīvā.

3.6.21. ROJ_Twitter_Object

ROJ_Twitter_Object klase glabā datus par *twitter* ziņām.

Mainīgo dokumentācija

- NSString* text – ziņas virsraksts.
- NSString* date – ziņas publicēšanas datums.

3.6.22. ROJ_Twitter_Cell, ROJ_Jaunumi_Cell

Šīs abas klases ir jaunumu sadaļas tabulas skata rindas elementi, kuri sastāv attiecīgi no jaunuma ziņām vai *twitter* ziņām.

4. PROJEKTA ORGANIZĀCIJA

Projekts tika izstrādāts pēc spējās programmatūras izstrādes metodes. Spējās programmatūras izstrādes metodes galvenā īpašība ir regulāru prasību izveide un esošo prasību pilnveidošana vai labošana. Šīs metode ir vairāk piemērota mazākiem projektiem, jo tiek ātrāk veikta programmatūras izstrāde un ir vieglāka problēmu risināšana un izmaiņu veikšana, kas rodas izstrādes brīdī.

Projekta organizēšanā tika izmantots projektu vadības rīks – *JIRA*[JIRA2013], ar kuru varēja sekot līdzi izdarītajiem uzdevumiem un plānot turpmākos uzdevumus. Šis rīks ļoti palīdzēja prognozēt lietotnes kopējo izstrādes termiņu.

Darba sākumā tika formulētas sistēmas prasības pēc klientu vēlmēm.

Pēc katras iterācijas tika iegūts strādājošs produkts un veikta produkta testēšana, lai nodrošinātu to, ka produkts izpilda izveidotos akcepttestus.

Kad tika iegūts galaprodukts, bija nepieciešams veikt visas sistēmas funkcionalitātes un veiktspēju testēšanu pirms tās publicēšanas.

Darba autors konsultējās ar darba vadītāju, bet darba plānošanā – ar uzņēmuma *iOS* programmētāju, kurš sniedza konsultācijas labāku metožu izvēlē. Projekta laikā tika organizētas darba komandas sapulces, kurās tika pārrunāts izdarītais un plānotas tālākās darbības. Kopumā darba komanda sastāvēja no 4 cilvēkiem – projekta vadītāja, *iOS* programmētāja, *Android* programmētāja un *PHP* programmētāja.

Autors veica sekojošus programmas izstrādes etapus:

- *iOS* programmēšanu,
- dokumentēšanu,
- lietotnes testēšanu un dokumentēšanu.

5. KONFIGURĀCIJAS PĀRVALDE

Programmatūras versiju kontrolei tika izmantota *SVN* pārvaldības sistēma. *SVN* ir centralizēta informācijas apmaiņas sistēma. Sistēmas kodols ir repozitorijs, kurš ir datu centrālā noliktava. Repozitorijam var piekļūt ar *HTTP* vai *HTTPS* savienojumu. *SVN* tika izvēlēts, jo to ir ērti un vienkārši lietot [SVN2004]. Uzņēmuma darbiniekiem ir lielāka *SVN* izmantošanas pieredze salīdzinājumā ar citām versiju kontroles sistēmām, tāpēc tas bija viens no argumentiem, kāpēc tika izvēlēta *SVN*. Tas ļauj jebkuram komandas dalībniekam labot kopējo projektu, nevis veidot katram savu izstrādes kopiju. *SVN* nestrādā lokāli, kas ir tā lielākais mīnuss. Izstrāde tika veikta darba vidē, kurā ir pieejams internets, tāpēc tas nesagādāja grūtības.

Pēc katra uzdevuma izpildīšanas un testēšanas, tika saglabātas izmaiņas sistēmā, kas ļauj viegli kontrolēt izmaiņas un sekot tām. Izmaiņas tika saglabātas ne retāk kā reizi dienā.

6. KVALITĀTES NODROŠINĀŠANA

Lai nodrošinātu augstu kvalitātes līmeni, tika veikta detalizēta testēšana pēc katras iterācijas un veikti akcepttesti. Pēc katras kļūdas izlabošanas testēšana tika veikta atkārtoti. Pēc produkta izstrādes, tika veikta lietotnes veiktspējas testēšana. Tajā tika izmantotas 3 metodes: *Xcode* analīze, *Xcode* „Instruments" un „Apple" servisa analīze.

Izmantojot *Xcode* analīzi, izstrādes vide veica kodu pārskatu un paziņoja par būtiskām kļūdām un iespējamiem draudiem lietotnes izmantošanā.

Xcode "Instruments" piedāvā daudzus lietotnes veiktspējas testus. Tika izvēlēti 4 testi:

- Atmiņas izmantošanas tests,
- Nepareizas atmiņas izmantošanas tests,
- Laika patēriņa tests,
- Enerģijas patēriņa tests.

„Apple" servisa analīze tiek izmantota, kad lietotne ir publicēta *AppStore*. Kļūdu gadījumos, lietotājs var nosūtīt kļūdu paziņojumu *AppStore*, kas pāradresē to izstrādātājiem.

Lai nodrošinātu kvalitatīvu kodu, tika veikta kodu bloku komentēšana un ievēroti „Apple" izstrādātie labās prakses *Objective – C* valodas ieteikumi. Viens no svarīgākajiem labās prakses ieteikumiem ir metožu un mainīgo deklarēšana pilnībā. Tas nozīmē, ka pēc nosaukuma var saprast, ko metode dara, kādi parametri tiek padoti un kas tiek atgriezts. Piemēram,*(UIImage*)imageForCellAtIndex:(NSIndexPath*)indexPathobject:(ROJ_Sadala_XMLObject*)object*. Tika ievērots, ka klases nosaukumi tika sākti ar lielo burtu, bet metodes – ar mazo.

Lietotne „Roja Travel Guide" tika izstrādāta pēc *MVC* modeļa principiem.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

Veicot programmatūras izstrādi pēc spējās programmatūras izstrādes pieejas, darbietilpības novērtējums ir balstīts uz izstrādātāja personīgo pieredzi konkrētā uzdevuma veikšanai. Pārsvarā spējās programmatūras izstrādes pieejā darbam veltītais laiks tiek definēts ar sarežģītības punktiem, kur viens punkts definē nepieciešamo laiku konkrēta uzdevuma veikšanai. Katram uzdevumam tiek piešķirti sarežģītības punkti, kas apzīmē uzdevuma darbietilpību.

Projektā viens sarežģītības punkts tika uzskatīts par vienu darba dienu. Tad tika vērtēta katra iterācija pēc tās piešķirtajiem uzdevumiem.

7.1.1. tabula

Katras iterācijas izstrādes dienu skaits pirms izstrādes

1.iterācija	11 dienas
2.iterācija	24 dienas
3.iterācija	10 dienas
4.iterācija	9 dienas
5.iterācija	2 dienas
6.iterācija	6 dienas

Tika plānots projekts izstrādāt 62 darba dienu laikā, bet tas tika pabeigts 58 darbu dienu laikā. Projekta izstrāde iekļāvās noteiktajā termiņā ar mazu rezerves laiku, kas tika atvēlēts lielāku kļūdu labošanai un veiktspējas uzlabošanai.

7.1.2. tabula

Katras iterācijas izstrādes dienu skaits pēc izstrādes

1.iterācija	10 dienas
2.iterācija	22 dienas
3.iterācija	9 dienas
4.iterācija	9 dienas
5.iterācija	2 dienas
6.iterācija	6 dienas

8. TESTĒŠANAS DOKUMENTĀCIJA

8.1. Akcepttesti

Visas programmatūras izstrādes laikā tika veikta vienībtestēšana un katram lietotājstāstam tika sastādīti akcepttesti.

Lai veiktu testēšanu, tika izmantotas 2 metodes:

- *Xcode* automatizētais testēšanas rīks,
- *NSLog()* funkcija.

Zemāk ir aplūkojamas izstrādātās akcepttestu tabulas.

8.1.1. tabula

1. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Lietotne tiek optimizēta darbībai iPhone versijai 4. Lietotnes simulācija pārbaudīta līdz iOS versijai 6. Lietotne darbojas, sākot no iOS versijas 5.0.	• Lietotne atveras uz iPhone.	• Lietotne atveras.
Lietotne tiek nodrošināta 2 valodās – latviešu un angļu valodā.	• Lietotne atveras uz iPhone. • Pēc lietotāja pieprasījuma tiek piedāvāta valodas maiņa.	• Lietotne atveras. • Uzrāda valodas maiņu skatu.
Valodas maiņa tiek veikta, izmantojot lietotnes iestatījumus.	• Lietotne atveras uz iPhone. • Lietotne nolasa informāciju no lokalizāciju datnēm. • Izvēlne nomaina valodu.	• Lietotne atveras. • Tiek iegūti vārdi no attiecīgās valodas lokalizācijas datnes. • Tiek nomainīta informācija un vārdi uz attiecīgo valodu.
Lietotnes dizains ir pielāgots visiem iPhone modeļu ekrāniem: • 320x480 pikseļi; • 640x960 pikseļi; • 640x1136 pikseļi.	• Lietotne atveras uz iPhone. • Lietotne nolasa attiecīgo dizaina datni.	• Lietotne atveras. • Uzrāda pareizās izšķirtspējas attēlus.

Zemāk var aplūkot 2. iterācijas akcepttestu tabulu, kas apraksta akcepttestus par kategorijas skatiem.

8.1.2. tabula

2. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Sākuma skatā tiek attēlotas 8 izvēles iespējas – Ko darīt, Ko redzēt, Kur palikt, Kur paēst, Ziņas, Informācija, Karte, Maršruti.	• Lietotne atveras uz iPhone. • Pēc lietotāja izvēles tiek atvērts attiecīgas kategorijas skats.	• Lietotne atveras. • Atveras attiecīgās kategorijas skats.
Sadaļu – Ko darīt, Ko redzēt, Kur palikt, Kur paēst, Maršruti – pirmais skats ietver objekta: • Attēlu, • Nosaukumu, • Attālumu no lietotāja līdz objektam. Ja ir iespējams, tad ir jāaprēķina attālums no objekta līdz lietotājam. Objekti sakārtoti pēc attāluma augošā secībā.	• Lietotne atveras uz iPhone. • Tiek apstrādāta attiecīgās kategorijas XML no servera. • Informācija tiek attēlota korekti. • Attēli tiek lejupielādēti paralēlā plūsmā. • Attālums tiek aprēķināts korekti.	• Lietotne atveras. • Iegūta informācija no servera. • Informācija tiek uzrādīta skatā. • Lietotne funkcionē kamēr tiek lejupielādēti attēli. • Tiek uzrādīts attālums.
Nospiežot uz objekta, par katru vietu tiek uzrādīts detalizēts apraksts un kontaktinformācija: • Nosaukums; • Attēls; • Preformatēts apraksts; • Kontaktinformācija (adrese, tālrunis, mob. tel., e–pasts, mājas lapa).	• Lietotne atveras uz iPhone. • Tiek uzrādīti korekti dati par konkrēto objektu. • Dati tiek izvadīti noteiktā kārtībā.	• Lietotne atveras. • Dati tiek uzrādīti skatā. • Dati ir sakārtoti.
Nospiežot uz kontaktinformācijas – telefona numuriem vai mājas lapām un attiecīgi automātiski atveras telefona zvanīšanas funkcija vai tīmekļa pārlūks.	• Lietotne atveras uz iPhone. • Uzspiežot uz tālruni, tiek atvērta zvanīšanas funkcija. • Uzspiežot uz saites, tiek atvērts pārlūks.	• Lietotne atveras. • Tiek zvanīts uz attiecīgo numuru. • Tiek atvērts pārlūks uz attiecīgo saiti.
Lietotājs var atstāt komentāru vai atsauksmi par konkrēto objektu. Ja ir norādīts objekta e–pasts, tad šo ziņu lietotne nosūta uz šo e–pastu, pretējā gadījumā, tā tiek nosūtīta uz kopējo e–pastu.	• Lietotne atveras uz iPhone. • Tiek nosūtīts pieprasījums uz serveri un atsauksmi nosūta uz konkrēto e–pasta adresi.	• Lietotne atveras. • Tiek nosūtīta vēstule uz konkrēto e–pasta adresi.

Zemāk var aplūkot 3. iterācijas akcepttestu tabulu, kas apraksta akcepttestus par jaunumu sadaļu.

8.1.3. tabula

3. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Jaunumi un aktualitātes ir jāuzrāda vizuāli baudāmā formā. Datu iegūšanai tiek izmantots AmberKit platformai pieejamais XML, kas satur: • Saiti uz raksta oriģinālo avotu; • Preformātētu aprakstu; • Attēlu; • Publicēšanas datumu.	• Lietotne atveras uz iPhone. • Tiek apstrādāts jaunumu XML no servera. • Tiek uzrādīta korekta informācija.	• Lietotne atveras. • Dati ir korekti apstrādāti. • Informācija tiek uzrādīta skatā.
Pārskatāmā veidā attēlots pašvaldības *twitter* konts, izvadot pēdējos 10 ierakstus. Tiek norādīts ieraksta publicēšanas datums un teksts.	• Lietotne atveras uz iPhone. • Tiek apstrādāts *twitter* XML no *twitter* servera. • Tiek izvadīti pēdējie 10 ieraksti. • Tiek uzrādīta korekta informācija.	• Lietotne atveras. • Dati ir korekti apstrādāti. • Skats satur 10 ierakstus. • Informācija tiek uzrādīta skatā.

Zemāk var aplūkot 4. iterācijas akcepttestu tabulu, kas apraksta akcepttestus par kartes un informācijas skatiem.

8.1.4. tabula

4. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Kartē, kurā pa atbilstošajām kategorijām – Ko darīt, Ko redzēt, Kur palikt, Kur paēst – ir parādītas apmeklējamās vietas, ja pie objekta ir norādītas koordinātes. Iespējams skatīties visas kategorijas vai kādas no tām atspējot.	• Lietotne atveras uz iPhone. • Lietotne atver karšu sistēmu. • Uz kartes tiek uzrādīti objekti. • Katrs objekts uzrāda tā nosaukumu un adresi (ja tāda ir norādīta). • No katra objekta var nokļūt objekta detalizētajā skatā. • Tiek korekti	• Lietotne atveras. • Atveras skats ar karšu sistēmu. • Objekti tiek attēloti uz kartes. • Uzspiežot uz objekta, parādās tā informācija. • Tiek uzrādīts objekta detalizētais skats. • Lietotne uzrāda tikai atlasītās

	atspējotas kategorijas.	kategorijas objektus.
Ir jāizveido informācijas lauks, kurā tiek attēlota bilde un pašvaldības iepriekš sagatavots teksts vizuāli baudāmā formā.	• Lietotne atveras uz iPhone. • Teksts tiek iegūts no datnes, un informācija tiek uzrādīta korekti.	• Lietotne atveras. • Informācija tiek uzrādīta skatā.

Zemāk var aplūkot 5. iterācijas akcepttestu tabulu, kas apraksta akcepttestus par *Google Analytics* statistikas izveidi.

8.1.5. tabula

5. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Lietotnei ir jānodrošina *Google Analytics* statistika.	• Lietotne atveras uz iPhone. • Tiek nosūtīti pieprasījumi uz *Google analytics* sistēmu.	• Lietotne atveras. • *Google analytics* uzrādās dati par lietotni.

Zemāk var aplūkot 6. iterācijas akcepttestu tabulu, kas apraksta akcepttestus par izstrādi, kas nodrošina lietotnes izmantošanu bezsaistes režīmā.

8.1.6. tabula

6. Iterācija

Lietotājstāsts	Akcepttests	Vēlamais rezultāts
Lietotnei ir jāstrādā bezsaistes režīmā (visi dati jāglabā uz ierīces)	• Lietotne atveras uz iPhone. • Bezsaistes gadījumā tiek izmantoti saglabātie dati. • Tiek nolasīti dati no ierīces datu glabātuves. • Pēc klientu datu korekciju veikšanas, dati tiek lejupielādēti un saglabāti uz ierīces.	• Lietotne atveras. • Ja nav pieejams interneta savienojums, tiek uzrādīta informācija no saglabātajiem datiem. • Tiek uzrādīta korekta informācija. • Veic datu lejupielādi pēc datu izmaiņu veikšanas.

8.2. Lietotnes „Roja Travel Guide” veiktspēja

Akcepttesti ir viens no veidiem, kā nodrošināt, ka lietotne strādā korekti. Arī lietotnes veiktspējas testi ir ļoti svarīgi. Lietotnes veiktspējas testi tika veikti lietotnes izveides pēdējā stadijā – pēc gala produkta izveides.

Lietotnes veiktspēja tika sadalīta trīs posmos:

- *Xcode* analīze,
- *Xcode* „Instruments” analīze,
- „Apple” servisa analīze.

8.2.1. *Xcode* analīze

Veiktspējas testa pirmais posms bija *Xcode* analīzes tests. Tas ir *Xcode* vides tests, kas izceļ būtiskākās kļūdas, kuras uzskatāmas par draudiem lietotnes veiktspējai. Testa rezultātā tika atklātas 8 kļūdas, kuras pārsvarā bija mainīgo deklarēšana, kuri citviet programmā netika izmantoti. *Xcode* analīze nenodrošina 100% kļūdu identificēšanu, tāpēc ir nepieciešams izmantot *Xcode „*Instruments*”* analīzi, kura veic detalizētākus testus.

8.2.2. *Xcode* „Instruments”

Xcode „Instruments” piedāvā dažādus testus. Šī kvalifikācijas darba ietvaros tika izmantoti 4 testi:

- Atmiņas izmantošanas tests;
- Nepareizas atmiņas izmantošanas tests;
- Laika patēriņa tests;
- Enerģijas patēriņa tests.

Atmiņas izmantošanas tests veic aprēķinus, cik daudz atmiņas tiek patērēts, veicot konkrētas lietotnes darbības. Tā kā lietotne izmanto *Xcode ARC* sistēmu, tad tika kontrolēts tas, ka atmiņas izmantošana būs normas robežās. Izpildītais atmiņas izmantošanas tests bija veiksmīgs.

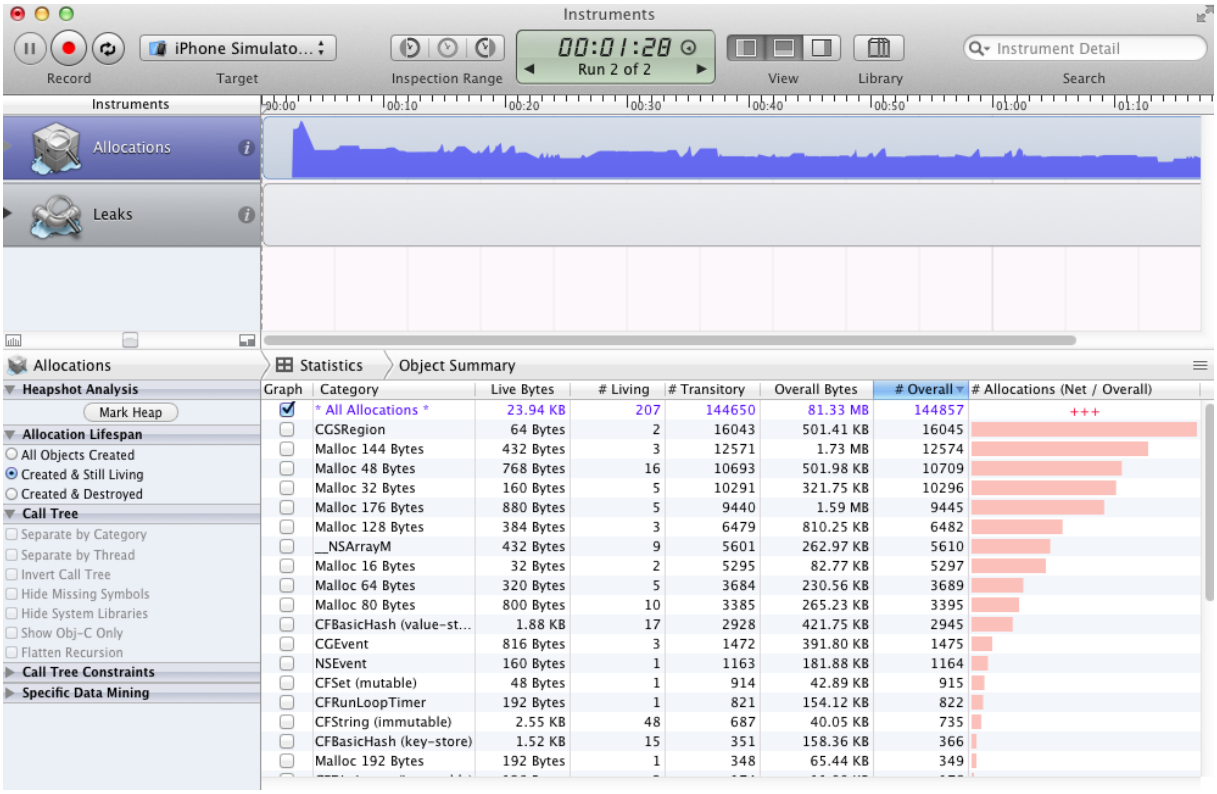
Nepareizas atmiņas izmantošanas tests uzrāda koda blokus, kuros objektam ir atvēlēts atmiņas apgabals, bet uz objektu nenorāda neviens rādītājs. Veicot testu, netika uzrādīts neviens koda bloks, kurā būtu novērota nepareizas atmiņas izmantošana. Izpildītais nepareizas atmiņas izmantošanas tests bija veiksmīgs.

Laika patēriņa tests uzrāda laiku, cik ir nepieciešams, lai veiktu lietotnes konkrētu darbību. Šī testa ietvaros tika izmantots *iPad 1* modelis, kuram ir vieni no zemākajiem sistēmas parametriem, kas atbalsta *iOS 5* versiju. Visilgākais laiks tika atzīmēts pie lietotnes atvēršanas. Pārsvarā skatu maiņa tika veikta ~ 1,5 sekundes ilgi.

Enerģijas patēriņa tests aprēķina katras metodes patērēto enerģijas daudzumu, kā arī kādas iekārtas komponentes tiek izmantotas konkrētas metodes veikšanai. Šis tests uzrāda vizuālu rezultātu, atverot kādu konkrētu skatu.

8.2.3. „Apple" servisa analīze

„Apple" servisa analīze ir pēdējais tests, kura ilgums ir visas lietotnes pastāvēšanas laiks. Uzņēmums nodrošina lietotnes testēšanu pirms tā tiek publicēta *AppStore* un pēc veiksmīgas publicēšanas. Lietotnes sabrukuma gadījumā „Apple" nodrošina ziņojumu saņemšanu no lietotājiem.



8.2.1 att. **Nepareizas atmiņas izmantošanas tests**

REZULTĀTI

Kvalifikācijas darba laikā tika izveidota Rojas novada tūrisma *iOS* lietotne „Roja Travel Guide”. Spējā programmatūras metode bija piemērota konkrētā projekta izveidei, jo, izmantojot lietotājstāstus un izveidotās iterācijas, autoram bija ērti organizēt darbu un veikt korekcijas pēc sanāksmēm ar klientiem.

Izstrādes procesu laika gaitā autors ieguva jaunas zināšanas par spējās programmatūras izstrādes principiem un projektu izstrādes gaitu, *Xcode* testēšanas rīkiem, kā arī par programmatūras izstrādes dokumentēšanu. Tika uzlabotas jau esošās zināšanas par *Objective – C* valodu, konfigurācijas pārvaldību, *iOS* programmatūras izstrādes principiem un *Mac OS X* operētājsistēmu.

Lietotne „Roja Travel Guide” ir pabeigta un drīzuma tiks publicēta *AppStore*.

SECINĀJUMI

Pēc lietotnes izstrādes autors secināja, ka dažas izdarītās izvēles būtiski uzlaboja lietotnes kvalitāti un izstrādes procesu. Izvēlētais programmatūras izstrādes modelis uzlaboja izstrādes organizēšanu, prasību apkopošanu pēc klientu vēlmēm un ērtāku izmaiņu veikšanu. Izstrādes gaitā tika veiktas daudzas sākotnējo prasību izmaiņas, tāpēc spējā programmatūras izstrādes metode bija ļoti piemērota, jo tā paredz ērtu izmaiņu veikšanu izstrādes gaitā. Tā kā spējās programmatūras izstrāde metode paredz regulāru tikšanos ar klientiem, tad tika novērstas vairākas domstarpības un nesaprašanās izstrādes gaitā. Lietotnes gala rezultāts atbilda klientu vēlmēm. Izvēlētās programmēšanas valodas un versiju kontroles sistēmas lietošana būtiski atviegloja izstrādes procesu. Klienti bija apmierināti ar gala rezultātu un darbības gaitu.

Izstrādes laikā autors saskārās ar vairākām *iOS* lietotņu izstrādes problēmām. Visbūtiskākā problēma bija vairāku pavedienu *(Multi–Threading)* nodrošināšana, kas saturēja vairākas problēmas. Viena no šīm problēmām bija savstarpēju pavedienu komunikācija un vienlaicīgu objektu izmantošana katrā pavedienā. Tāpēc bija nepieciešams veikt daudzas drošības darbības, kas novērstu lietotnes darbības sabrukumu. Salīdzinot ar vecākām *iOS* versijām (3.0), tad vairāku pavedienu *(Multi–Threading)* sistēma ir būtiski uzlabota, lai novērstu šādas problēmas.

Autors secina, ka kvalifikācijas darba izstrādes laikā ieguva lielu pieredzi mobilo lietotņu izstrādē, programmatūras dokumentēšanā un mobilo tehnoloģiju lietošanā. Izstrādātā lietotne atbilst iecerētajam produktam un klientu vēlmēm, kas drīzumā tiks publicēta *AppStore*. Tā kā mobilās tehnoloģijas attīstās ļoti strauji un katra nākamā *iOS* versija sniedz plašākas funkcionalitātes iespējas, tad nākotnē plānots ieviest papildus funkcionalitāti un uzlabot esošo, kas veicinātu lietotnes pieprasījumu.

IZMANTOTĀ LITERATŪRA

[Adriaenssen2013] **Adriaenssen Tom** *ViewDeck* [tiešsaiste] – [atsauce 28.04.2013.]. Pieejams: *https://github.com/Inferis/ViewDeck*

[Apple2013a] **„Apple"** *What's new in iOS – iOS 5.0* [tiešsaiste] – [atsauce 29.03.2013.]. Pieejams: *http://developer.apple.com/library/ios/#releasenotes/General/WhatsNewIniOS/Articles/iOS5.html*

[Apple2013b] **„Apple"** *iOS Dev center* [tiešsaiste] – [atsauce 05.04.2013.]. Pieejams: *https://developer.apple.com/devcenter/ios/index.action*

[Apple2013c] **„Apple"** *Model–View–Controller* [tiešsaiste] – [atsauce 24.03.2013.]. Pieejams: *http://developer.apple.com/library/ios/#documentation/general/conceptual/devpedia–cocoacore/MVC.html*

[Bukovinski2013] **Bukovinski Matej** *MBProgressHUD* [tiešsaiste] – [atsauce 25.04.2013.]. Pieejams: *https://github.com/jdg/MBProgressHUD*

[JIRA2013] **„Atlassian"** *JIRA* [tiešsaiste] – [atsauce 02.06.2013.]. Pieejams: *http://www.atlassian.com/software/jira*

[Mark2011] **Mark David, Nutting Jack, LaMarche Jeff** *Beginning iOS 5 Development: Exploring the iOS SDK*. New York : Apress, 2011 , 217–277lpp.

http://books.google.lv/books/about/Beginning_iOS_5_Development.html?id=sikKz–oQNFIC&redir_esc=y

[Million2013] **Million Tony** *Reachability* [tiešsaiste] – [atsauce 16.04.2013.]. Pieejams: *https://github.com/tonymillion/Reachability*

[Ramanathan2012] **Ramanathan, Valli, Meenakshi.** *Worldwide Smartphone Users Cross 1 Billion Mark: Report*. [tiešsaiste] – [atsauce 19.03.2013.]. Pieejams: *http://www.ibtimes.com/worldwide–smartphone–users–cross–1–billion–mark–report–847769*

[Sadun2012] **Sadun Erica** *The iOS 5 Developer's Cookbook: Core Concepts and Essential Recipes for iOS Programmers.* Boston: Addison–Wesley Professional, 2012 , 3. sējums, 695–745lpp.

http://books.google.lv/books/about/The_IOS_5_Developer_s_Cookbook.html?id=yUD–PZJsFtEC&redir_esc=y

[Singh2013] **Singh, Sameer.** *How Many iPhones Apple Will Sell Around The World* [tiešsaiste] – [atsauce 18.03.2013.]. Pieejams: *http://www.businessinsider.com/iphone–5–sales–by–area–around–the–world–2012–9*

[Smith2013] **Smith David** *iOS Version Stats* [tiešsaiste] – [atsauce 27.03.2013.]. Pieejams: *http://david–smith.org/iosversionstats/*

[SVN2004] *Versiju kontrole ar Subversion un Git* [tiešsaiste] – [atsauce 10.03.2013.]. Pieejams: *https://www.google.lv/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CEYQFjAD&url=http%3A%2F%2Ffiles–ante–lv.googlecode.com%2Fsvn%2Ftrunk%2FTraining.WebProgramming.Masters2012%2Fpresentations%2FLekcija01%2520–%252004_VersionControl_2012–09–06.ppt&ei=jguqUarFNMaQ4gS9z4CYCg&usg=AFQjCNE9Us28RxNszB7NdzAXLMtdxL54Dw&sig2=heVRpMNtlB2locaUGhhlQg*

[thenewboston.org2013] **thenewboston.org** *Objective–C* [tiešsaiste] – [atsauce 09.04.2013.]. Pieejams: *http://thenewboston.org/list.php?cat=33*

PIELIKUMS

1. pielikums. Lietotnes „Roja Travel Guide” koda fragmenti

Zemāk var aplūkot lietotnes koda fragmentu no pamat skata klases.

```
//
//  ROJ_SadalaMainViewController.m
//  Roja
//
//  Created by Armands Baurovskis on 1/23/13.
//  Copyright (c) 2013 AmberPhone. All rights reserved.
//

#import "ROJ_SadalaMainViewController.h"

@interface ROJ_SadalaMainViewController ()
{
    CLLocationManager * locationManager;
    MBProgressHUD * HUD;
    NSUserDefaults * defaults;
    int currenntLanguage;
    ROJ_Text_Localized * textLocalization;
    ConnectionFinder * connection;
    NSString * language;
    int buttonID;
}
@end

@implementation ROJ_SadalaMainViewController

@synthesize
infoFromServer=_infoFromServer,labelToDo=_labelToDo,labelToStay=_labelToStay,labelToSee=_la
belToSee,labelToEat=_labelToEat,labelNews=_labelNews,labelRoute=_labelRoute,labelMap=_label
Map,labelInfo=_labelInfo,novadaText=_novadaText;

– (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
```

```
/*  Method which checks if passed version is the same which is on device
 version – version number ( 5.0 , 5.1 etc)
 Return YES or NO , if version matches */
–(BOOL)systemVersion:(NSString*)version
{
   //Compare if system version is equal to given version
   if([[[UIDevice currentDevice] systemVersion] compare:version options:NSNumericSearch] ==
NSOrderedSame)
   {
      return YES;
   }
   else
   {
      return NO;
   }
}
–(void)viewWillAppear:(BOOL)animated
{
   defaults = [NSUserDefaults standardUserDefaults];
   connection= [[ConnectionFinder alloc]init];
   language = [defaults objectForKey:@"valoda"];

   //Get langaunge identificator
   if([language isEqualToString:@"lv–LV"])
   {
      currenntLanguage=textLocalization.language=1;
   }
   else if([language isEqualToString:@"en"])
   {
      currenntLanguage=textLocalization.language=2;
   }

   //Change label texts and images for specific languange from Localizable.strings file
   [self changeLabelTexts];

   //Remove sidebar menu from view
   self.viewDeckController.rightController=nil;
   //Remove navigation bar from view
   self.navigationController.navigationBarHidden=YES;

}
/* Method which changes all view labels text for specific languange */
–(void)changeLabelTexts
{
   //Change label texts and images for specific languange from Localizable.strings file
   _novadaText.text=[[textLocalization keyForLanguage:@"DISTRICT"]uppercaseString];
   _labelToDo.text=[[textLocalization keyForLanguage:@"LABEL_DO"]uppercaseString];
   _labelToStay.text=[[textLocalization keyForLanguage:@"LABEL_STAY"]uppercaseString];
   _labelInfo.text=[[textLocalization keyForLanguage:@"LABEL_INFO"]uppercaseString];
   _labelToEat.text=[[textLocalization keyForLanguage:@"LABEL_EAT"]uppercaseString];
   _labelToSee.text=[[textLocalization keyForLanguage:@"LABEL_SEE"]uppercaseString];
   _labelNews.text=[[textLocalization keyForLanguage:@"LABEL_NEWS"]uppercaseString];
   _labelRoute.text=[[textLocalization keyForLanguage:@"LABEL_ROUTE"]uppercaseString];
   _labelMap.text=[[textLocalization keyForLanguage:@"LABEL_MAP"]uppercaseString];
}
```

```
/* Get user's selected languange  sender – pressed button's variable */
– (IBAction)getLanguage:(id)sender {

    //Change image and texts when new languange is loaded
    switch ([sender tag]) {
        case 0:
        {
            //Set default user languange in UserDefaults DB
            defaults = [NSUserDefaults standardUserDefaults];
            [defaults setObject:@"lv–LV" forKey:@"valoda"];
            [defaults synchronize];

            //Set values to Latvian
            language=@"lv–LV";
            currenntLanguage=1;
            textLocalization.language=1;

            //Change label texts and images for specific languange from Localizable.strings file
            [self changeLabelTexts];

            //Hide langunage view with fade effect
            [self hideLangaungeView];
            NSLog(@"CHANGED LANGUANGE SUCCESSFULL");

            //Show activity indicator
            HUD = [[MBProgressHUD alloc] initWithView:self.view];
            [self.view addSubview:HUD];
            HUD.delegate = self;
            [HUD show:YES];

            //Get new data
            [NSThread detachNewThreadSelector:@selector(receiveData) toTarget:self withObject:nil];
        }
            break;
        case 1:
        {
            //Set default user languange in UserDefaults DB
            defaults = [NSUserDefaults standardUserDefaults];
            [defaults setObject:@"en" forKey:@"valoda"];
            [defaults synchronize];

             //Set values to Englinsh
            language=@"en";
            textLocalization.language=2;
            currenntLanguage=2;

             //Change label texts and images for specific languange from Localizable.strings file
            [self changeLabelTexts];

            //Hide langunage view with fade effect
            [self hideLangaungeView];
            NSLog(@"CHANGED LANGUANGE SUCCESSFULL");
```

```
            //Show activity indicator
            HUD = [[MBProgressHUD alloc] initWithView:self.view];
            [self.view addSubview:HUD];
            HUD.delegate = self;
            [HUD show:YES];
            //Get new Data
            [NSThread detachNewThreadSelector:@selector(receiveData) toTarget:self withObject:nil];
        }
            break;
        default:
            break;
    }
}
–(void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    NSLog(@"LOADING ANOTHER VIEWCONTROLLER WAS SUCCESSFULL");
}
/* Hide with animation languange view with fade effect*/
–(void)hideLangaungeView
{
    //Using CoreAnimations , hide view using fade out effect
    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:0.5];
    [self.viewValodas setAlpha:0.0];
    [UIView commitAnimations];
    NSLog(@"HIDE LANGUANGE VIEW SUCCESFULL");
}
/* Hide langaunge view after button is pressed */
– (IBAction)hideView:(id)sender {
    [self hideLangaungeView];
}
/* Notification method, which pops to root controller and go to specific view */
–(void)messageFromMenu
{
    //Pop to Root Controller
    [self.navigationController popToRootViewControllerAnimated:NO];
    //Remove sidemenu
    [self.viewDeckController toggleRightViewAnimated:YES];
    //Save user choice from sidemenu
    defaults = [NSUserDefaults standardUserDefaults];
    NSString * task = [defaults objectForKey:@"objekti"];
    //Show specific ViewController
    [self performSegueWithIdentifier:task sender:self];
}
/* Read data from DB  name – file name  Return array with data */
–(NSMutableArray*)readData:(NSString*)name
{
    //Get path to applications DB
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *docDir = [paths objectAtIndex:0];
    NSString *fullFileName = [NSString stringWithFormat:@"%@/%@", docDir,name];
    //Get archive docomunent from DB with specific name
    NSMutableArray *arrayFromDisk = [NSKeyedUnarchiver unarchiveObjectWithFile:fullFileName];
    return arrayFromDisk;
}
```

```
/* Add data to DB name – file name  object – array which will be saved */
–(void)AddData:(NSMutableArray*)object :(NSString *)name
{
   //Get path to application DB
   NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
   NSString *docDir = [paths objectAtIndex:0];
   NSString *fullFileName = [NSString stringWithFormat:@"%@/%@", docDir,name];

   //Make archive document and save it in DB
   [NSKeyedArchiver archiveRootObject:object toFile:fullFileName];
}

/*Load new data , when languange is changed*/
–(void)receiveData
{
   //Receive data
   [self getData];
   //On main thread, remove activity indicator
   dispatch_async(dispatch_get_main_queue(), ^{
      [HUD hide:YES];
      NSLog(@"LOADED DATA SUCCESSFULL");
   });
}

– (void)viewDidLoad
{
   [super viewDidLoad];
   textLocalization=[[ROJ_Text_Localized alloc]init];
   connection= [[ConnectionFinder alloc]init];

   //Read user default languange from UserDefaults DB
   defaults = [NSUserDefaults standardUserDefaults];
   language = [defaults objectForKey:@"valoda"];

   //Add listeneres to NSnotificationCenter
   [[NSNotificationCenter defaultCenter] addObserver: self selector: @selector(messageFromMenu)
name:@"menuTask" object: nil];
   [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(receiveData)
name:@"SadalaMainViewController" object:nil];

   /* Change navigation controller color */
   UIColor * color = [UIColor colorWithRed:9/255.0f green:76/255.0f blue:145/255.0f alpha:1];
   self.navigationController.navigationBar.tintColor=color;

   //Start location manager which will get user current locations and will update them
   if (locationManager == nil)
   {
      locationManager = [[CLLocationManager alloc] init];
      locationManager.desiredAccuracy =
      kCLLocationAccuracyNearestTenMeters;
      locationManager.delegate = self;
   }
   [locationManager startUpdatingLocation];
```

```
    //Check if app is loaded for specific ios version
    if ([self systemVersion:@"5.0"]) {
        NSLog(@"5.0 Successful loading");
    }
    else if([self systemVersion:@"5.0.1"])
    {
        NSLog(@"5.0.1 Successful loading");
    }
    else if([self systemVersion:@"5.1"])
    {
        NSLog(@"5.1 Successful loading");
    }
    else if([self systemVersion:@"6.0"])
    {
        NSLog(@"6.0 Successful loading");
    }
    else if([self systemVersion:@"6.1"])
    {
        NSLog(@"6.1 Successful loading");
    }

        // Do any additional setup after loading the view.
}

– (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

/* Get data and parse them */
–(void)getData
{
    //Alloc XML parsr
    _infoFromServer = [[ROJ_Sadala_XMLParser alloc]initXMLParser];

    //Load data from server. If internet connection is available, load from server, if not then load from
storage.
    if([connection isConnectionAvailable]){
        NSURL * XMLUrl =[[NSURL
alloc]initWithString:@"http://roja.amberkit.amberphone.lv/objekti.xml"];
        if([language isEqualToString:@"en"])
        {
            XMLUrl =[[NSURL alloc]initWithString:@"http://roja.amberkit.amberphone.lv/objekti–
eng.xml"];
        }
        //Get data from specific URL
        NSData *XML = [[NSData alloc] initWithContentsOfURL:XMLUrl];

        //Parse data
        NSXMLParser *XMLParser = [[NSXMLParser alloc] initWithData:XML];
        [XMLParser setDelegate:_infoFromServer];
        [XMLParser parse];
```

```
        //Save data in DB
        if([language isEqualToString:@"lv–LV"]){
            [self AddData:_infoFromServer.objekti :@"LV"];
        }
        else if([language isEqualToString:@"en"])
        {
            [self AddData:_infoFromServer.objekti :@"ENG"];
        }
    }
    else
    {
        if([language isEqualToString:@"lv–LV"]){
            _infoFromServer.objekti=[self readData:@"LV"];
            //If the storage has no data, load xml from project bundle
            if(!_infoFromServer.objekti)
            {
                //Parse data from XML
                NSString* path = [[NSBundle mainBundle] pathForResource:@"objektiLV"
                                                      ofType:@"xml"];
                NSData * data2 = [NSData dataWithContentsOfFile:path];
                NSXMLParser *XMLParser = [[NSXMLParser alloc] initWithData:data2];
                [XMLParser setDelegate:_infoFromServer];
                [XMLParser parse];
            }
        }
        else if([language isEqualToString:@"en"])
        {
            _infoFromServer.objekti=[self readData:@"ENG"];
            //If the storage has no data, load xml from project bundle
            if(!_infoFromServer.objekti)
            {
                //Parse data from XML
                NSString* path = [[NSBundle mainBundle] pathForResource:@"objektiENG"
                                                      ofType:@"xml"];
                NSData * data2 = [NSData dataWithContentsOfFile:path];
                NSXMLParser *XMLParser = [[NSXMLParser alloc] initWithData:data2];
                [XMLParser setDelegate:_infoFromServer];
                [XMLParser parse];

            }
        }
    }
}
/* Load data for first time */
–(void)receiveDataFirstTime
{
    //Receive data
    [self getData];
    //On main thread, remove activity indicator
    dispatch_async(dispatch_get_main_queue(), ^{
        [HUD hide:YES];
        NSLog(@"LOADED DATA SUCCESSFULL");
    });
    //Show specific ViewController
    [self performSelectorOnMainThread:@selector(pushView) withObject:nil waitUntilDone:NO];

}
```

```
/* Push to specific ViewController */
–(void)pushView
{
    //Push to specific viewcontroller.
    switch (buttonID) {
        case 0:
            [self performSegueWithIdentifier:@"toDo" sender:self];
            break;
        case 1:
            [self performSegueWithIdentifier:@"route" sender:self];
            break;
        case 2:
            [self performSegueWithIdentifier:@"toStay" sender:self];
            break;
        case 3:
            [self performSegueWithIdentifier:@"toEat" sender:self];
            break;
        case 4:
            [self performSegueWithIdentifier:@"toSee" sender:self];
            break;
        case 5:
            [self performSegueWithIdentifier:@"toMap" sender:self];
            break;
        default:
            break;
    }
}
– (void)viewDidUnload {

    [self setLabelToDo:nil];
    [self setLabelToDo:nil];
    [self setLabelToStay:nil];
    [self setLabelRoute:nil];
    [self setLabelToStay:nil];
    [self setLabelToEat:nil];
    [self setLabelToSee:nil];
    [self setLabelInfo:nil];
    [self setLabelNews:nil];
    [self setLabelMap:nil];
    [self setViewValodas:nil];
    [self setNovadaText:nil];
    [super viewDidUnload];
}
–(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{


    //Pass data to next viewcontroller

    ROJ_Saraksts_ViewController *destViewController = segue.destinationViewController;
    if(_infoFromServer.objekti.count!=0){
    if ([segue.identifier isEqualToString:@"toDo"])
    {
        textLocalization.language=currenntLanguage;
        destViewController.dataFromSadala=[_infoFromServer.objekti objectAtIndex:0];
        destViewController.tips=[textLocalization keyForLanguage:@"LABEL_DO"] ;
```

```
    /*Track view for Google Analytics */
    id tracker1 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker1 sendView:@"Ko darīt"];
}
else if ([segue.identifier isEqualToString:@"toStay"])
{
    textLocalization.language=currenntLanguage;
    destViewController.dataFromSadala=[_infoFromServer.objekti objectAtIndex:1];
    destViewController.tips=[textLocalization keyForLanguage:@"LABEL_STAY"] ;

    /*Track view for Google Analytics */
    id tracker2 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker2 sendView:@"Kur palikt"];
}
else if ([segue.identifier isEqualToString:@"toSee"])
{
    textLocalization.language=currenntLanguage;
    destViewController.dataFromSadala=[_infoFromServer.objekti objectAtIndex:2];
    destViewController.tips=[textLocalization keyForLanguage:@"LABEL_SEE"] ;

    /*Track view for Google Analytics */
    id tracker3 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker3 sendView:@"Ko redzēt"];
}
else if ([segue.identifier isEqualToString:@"toEat"])
{
    textLocalization.language=currenntLanguage;
    destViewController.dataFromSadala=[_infoFromServer.objekti objectAtIndex:3];
    destViewController.tips=[textLocalization keyForLanguage:@"LABEL_EAT"] ;

    /*Track view for Google Analytics */
    id tracker4 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker4 sendView:@"Kur pāest"];
}
else if ([segue.identifier isEqualToString:@"route"])
{
    textLocalization.language=currenntLanguage;
    destViewController.tips=[textLocalization keyForLanguage:@"LABEL_ROUTE"];
    destViewController.dataFromSadala=[_infoFromServer.objekti objectAtIndex:4];

    /*Track view for Google Analytics */
    id tracker5 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker5 sendView:@"Maršruti"];
}
else if ([segue.identifier isEqualToString:@"toMap"])
{
    textLocalization.language=currenntLanguage;
    ROJ_Map_ViewController *destViewController = segue.destinationViewController;
    destViewController.coordinates=[_infoFromServer.objekti objectAtIndex:5];
    /*Track view for Google Analytics */
    id tracker6 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
    [tracker6 sendView:@"Karte"];
}
}
```

```
    else
    {
        if ([segue.identifier isEqualToString:@"toDo"])
        {
            textLocalization.language=currenntLanguage;
            destViewController.tips=[textLocalization keyForLanguage:@"LABEL_DO"];
            /*Track view for Google Analytics */
            id tracker1 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
            [tracker1 sendView:@"Ko darīt"];
        }
        else if ([segue.identifier isEqualToString:@"toStay"])
        {
            textLocalization.language=currenntLanguage;
                    destViewController.tips=[textLocalization keyForLanguage:@"LABEL_STAY"] ;
            /*Track view for Google Analytics */
            id tracker2 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
            [tracker2 sendView:@"Kur palikt"];
        }
        else if ([segue.identifier isEqualToString:@"toSee"])
        {

            textLocalization.language=currenntLanguage;
            destViewController.tips=[textLocalization keyForLanguage:@"LABEL_SEE"] ;
            /*Track view for Google Analytics */
            id tracker3 = [[GAI sharedInstance] trackerWithTrackingId :kGoogleCode];
            [tracker3 sendView:@"Ko redzēt"];
        }
        else if ([segue.identifier isEqualToString:@"toEat"])
        {
            textLocalization.language=currenntLanguage;
            destViewController.tips=[textLocalization keyForLanguage:@"LABEL_EAT"] ;
            /*Track view for Google Analytics */
            id tracker4 = [[GAI sharedInstance] trackerWithTrackingId: kGoogleCode];
            [tracker4 sendView:@"Kur paēst"];
        }
        else if ([segue.identifier isEqualToString:@"route"])
        {
            textLocalization.language=currenntLanguage;
            destViewController.tips=[textLocalization keyForLanguage:@"LABEL_ROUTE"] ;
            /*Track view for Google Analytics */
            id tracker5 = [[GAI sharedInstance] trackerWithTrackingId:kGoogleCode];
            [tracker5 sendView:@"Maršruti"];
        }
    }
}
/* Show languange view with fade in effect */
– (IBAction)setLanguage:(id)sender {
    //Using CoreAnimations, show languange view using fade in effect
    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:0.5];
    [self.viewValodas setAlpha:1.0];
    [UIView commitAnimations];
    NSLog(@"SHOW LANGUANGE VIEW SUCCESFULL");
}
```

```
/* Show erron on screen */
–(void)getError
{
   //Alloc HUD.
   MBProgressHUD *hud = [MBProgressHUD showHUDAddedTo:self.view animated:YES];

        // Configure for text only and offset down
        hud.mode = MBProgressHUDModeText;
   textLocalization.language=currenntLanguage;
   UIFont *Font = [UIFont fontWithName:@"Arial–BoldMT" size:14.0];
   hud.labelFont = Font;
  hud.labelText=[textLocalization keyForLanguage:@"LABEL_INTERNET_ERROR"] ;
        hud.margin = 10.f;
        hud.yOffset = 150.f;
        hud.removeFromSuperViewOnHide = YES;

   //Show error for 2 seconds
        [hud hide:YES afterDelay:2];
}
/*If connection is available, push to news viewcontroller. If not, then show error */
– (IBAction)goToNews:(id)sender {
   if([connection isConnectionAvailable])
   {
      [self performSegueWithIdentifier:@"news" sender:self];
   }
   else
   {
      //Show error
      [self getError];
   }
}

/* Register pressed option and get data if the array is empty */
– (IBAction)pressedOption:(id)sender {
   buttonID = [sender tag];
   //If there is no data – load them
   if(_infoFromServer.objekti.count==0)
    {
      //Show activity indicator
      HUD = [[MBProgressHUD alloc] initWithView:self.view];
      [self.view addSubview:HUD];
      HUD.delegate = self;
      [HUD show:YES];

      //Load data
      [NSThread detachNewThreadSelector:@selector(receiveDataFirstTime) toTarget:self
withObject:nil];
    }
   else
   {
      //Push to specific viewController
      [self pushView];
   }
}
@end
```

2. pielikums. AmberKit

Lietotne „Roja Travel Guide" un citu lietotņu viena no galvenajām funkcijām ir datu saņemšana no ārēja servera, tāpēc ir nepieciešams izstrādāt vadības sistēmu, kura ļautu ērti veikt datu apmaiņu ar lietotnēm. *AmberKit* ir lietotņu pārvaldīšanas sistēma, kuru ir izstrādājusi SIA „AmberPhone".

AmberKit ļauj ērti izveidot katrai lietotnei savu kontu un reģistrēt lietotājus lietotnes kontā un pārvaldīt lietotni. Katrai lietotnei tiek izstrādāta tīmekļa lapa atkarībā no nepieciešamām darbībām. Tāda tipa sistēmas izveide būtiski uzlabo lietotnes komunikāciju ar serveri. Šādu sistēmu būtu ērti un vienkārši izmantot jebkuram lietotājam.

AmberKit ļauj veikt šādas darbības:

- Veikt informācijas datu apmaiņu ar lietotni,
- Veikt *Push* pieprasījumus uz lietotni un saņemt statistikas datus par tiem,
- Saņemt un glabāt datus no lietotnes un veikt datu statistisku analīzi,
- Uzrādīt *Google Analytics* statistikas datus par lietotni,
- Sniegt mērķētus ziņojumus – akcijas, speciālos piedāvājumus.

Rojas novada tūrisma lietotnei ir svarīgi saņemt datus dinamiski par katras kategorijas objektiem – to aprakstu, attēlus, kontaktinformāciju u.c. *AmberKit* ļauj ērti papildināt informāciju par objektiem jebkurā laikā.

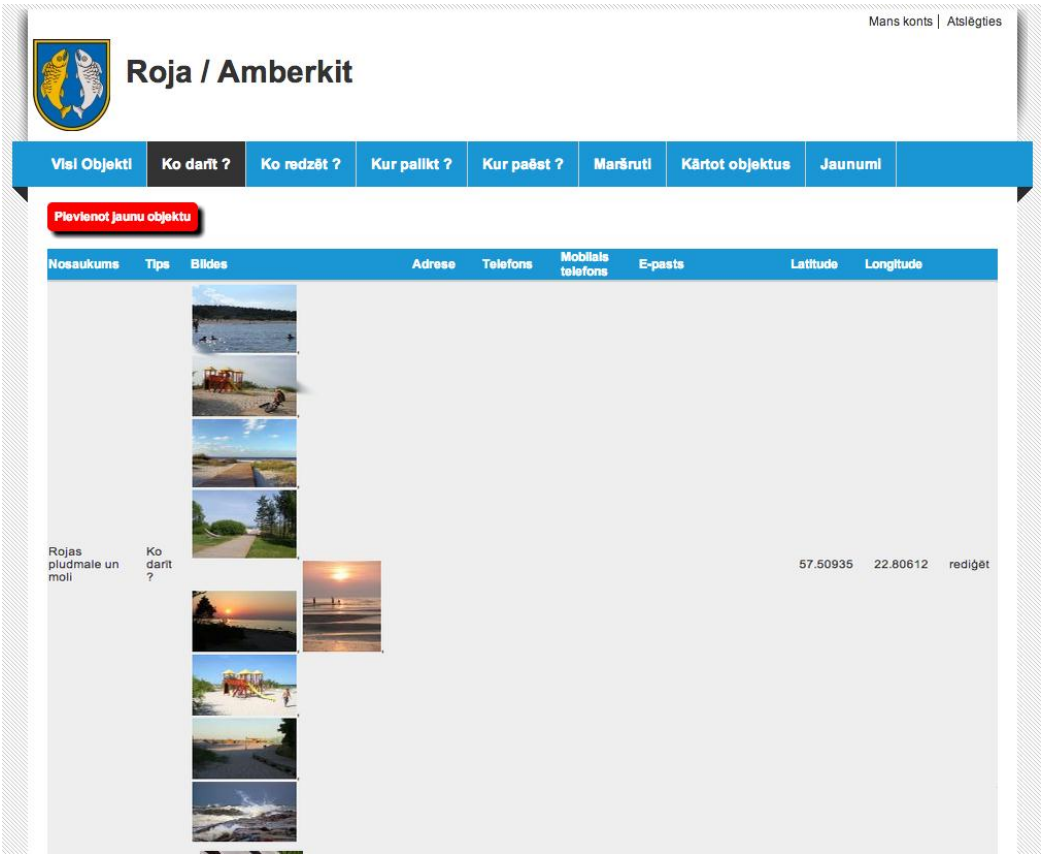
Izmantojot *AmberKit* kontu, ir iespējams nosūtīt *push* ziņas gan *iOS* lietotājiem, gan *Android* lietotājiem. Pēc tam tiek uzrādīta statistika par to, cik cilvēki ir saņēmuši ziņas un cik ir izlasījuši ziņu (nospieduši uz tās). *Push* ziņas pārsvarā tiek izmantotas kā mārketinga ziņas (piemēram, preču atlaides) vai lietotāja informēšanai par lietotnes izmaiņām.

Vairums lietotņu piedāvā iespēju sniegt atsauksmes gan par lietotni, gan par konkrētu servisu, tādēļ lietotne nosūta atsauksmes datus *AmberKit*, kas vēlāk veic to apstrādi un pārskatāmi uzrāda tās konta lietotājiem.

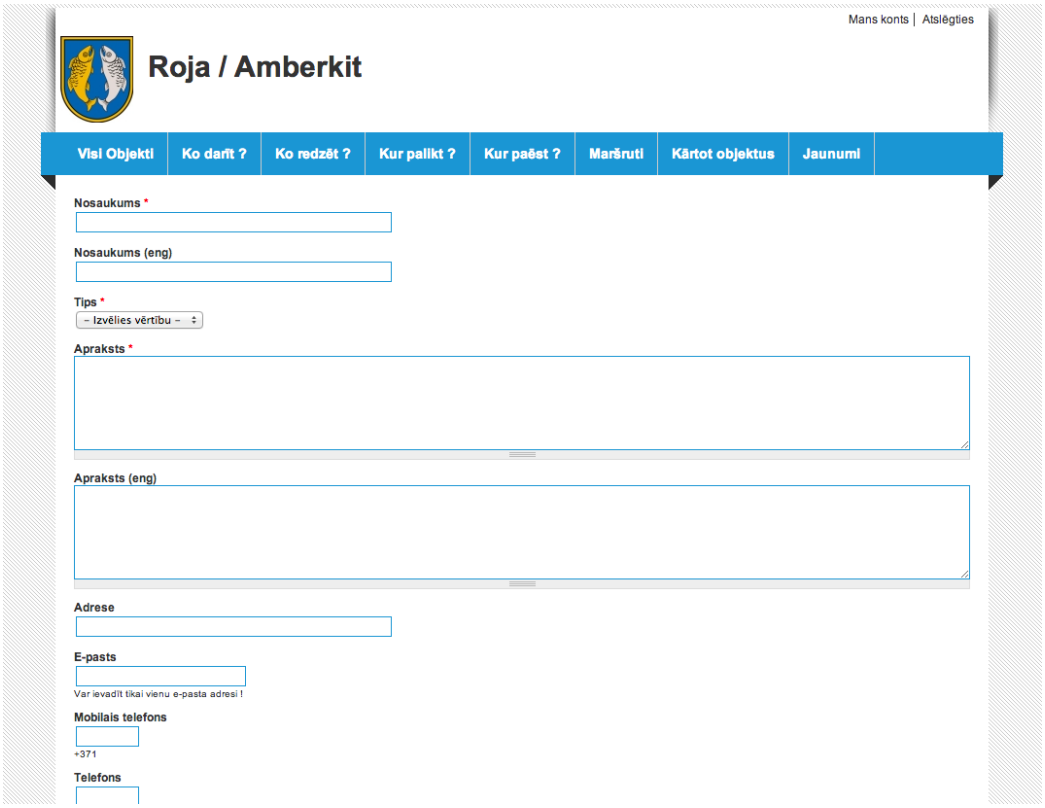
Google Analytics serviss piedāvā ļoti pārskatāmu statistiku par lietotnes izmantošanu – kuri skati tika apskatīti visvairāk, kuras valsts iedzīvotāji izmanto lietotni visvairāk u.c. *AmberKit* piedāvā pārskatāmu *Google Analytics* statistikas datu analīzi, kas ļauj konta lietotājiem iepazīties datiem par lietotnes izmantošanu.

Daudzas tirdzniecības veikalu lietotnes izmanto akciju un kuponu uzrādīšanu mārketinga veicināšanai. *AmberKit* ļauj veikt akciju pievienošanu, veikt to datu rediģēšanu, norādīt rādīšanas sākuma un beigu laiku u.c. informāciju. Pēc datu rediģēšanas lietotne veic atkārtotu datu lejupielādi un uzrāda aktuālāko informāciju.

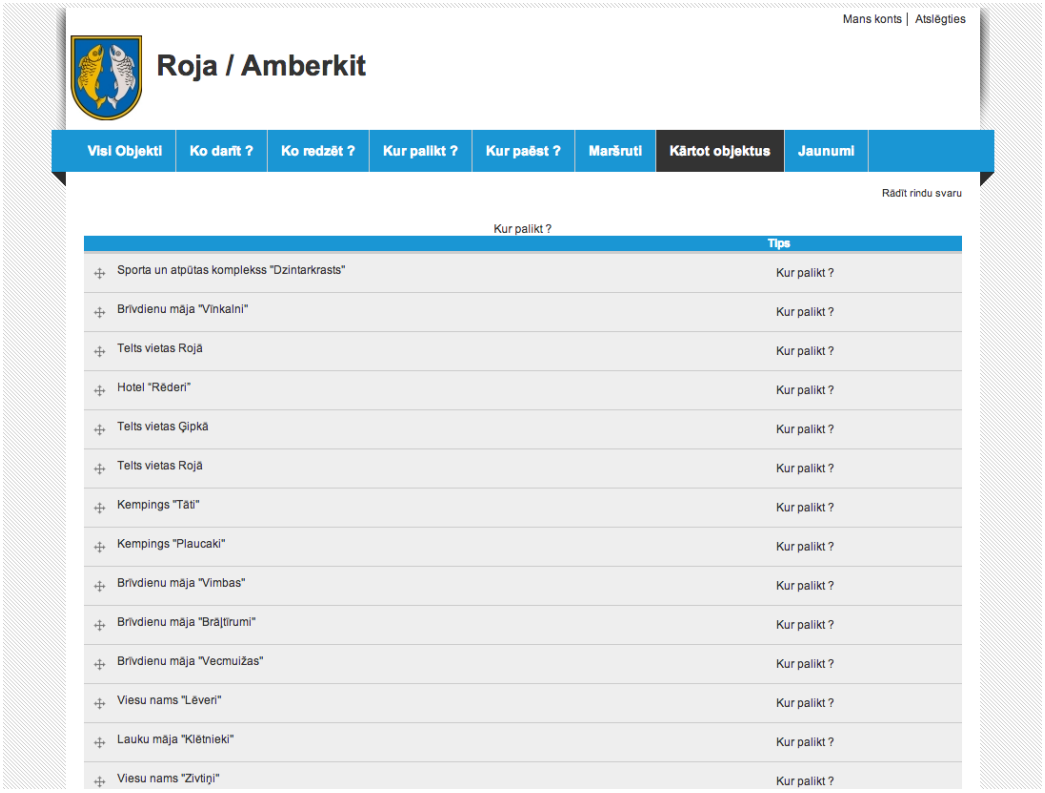
AmberKit būtiski uzlabo lietotnes iespējas un ļauj ērti un vienkārši veikt datu pievienošanu vai rediģēšanu un veikt citas ar lietotni saistītas darbības. *AmberKit* pašlaik tiek attīstīts un papildināts ar citām iespējām, kas būtiski paplašinātu lietotnes darbību spektru.



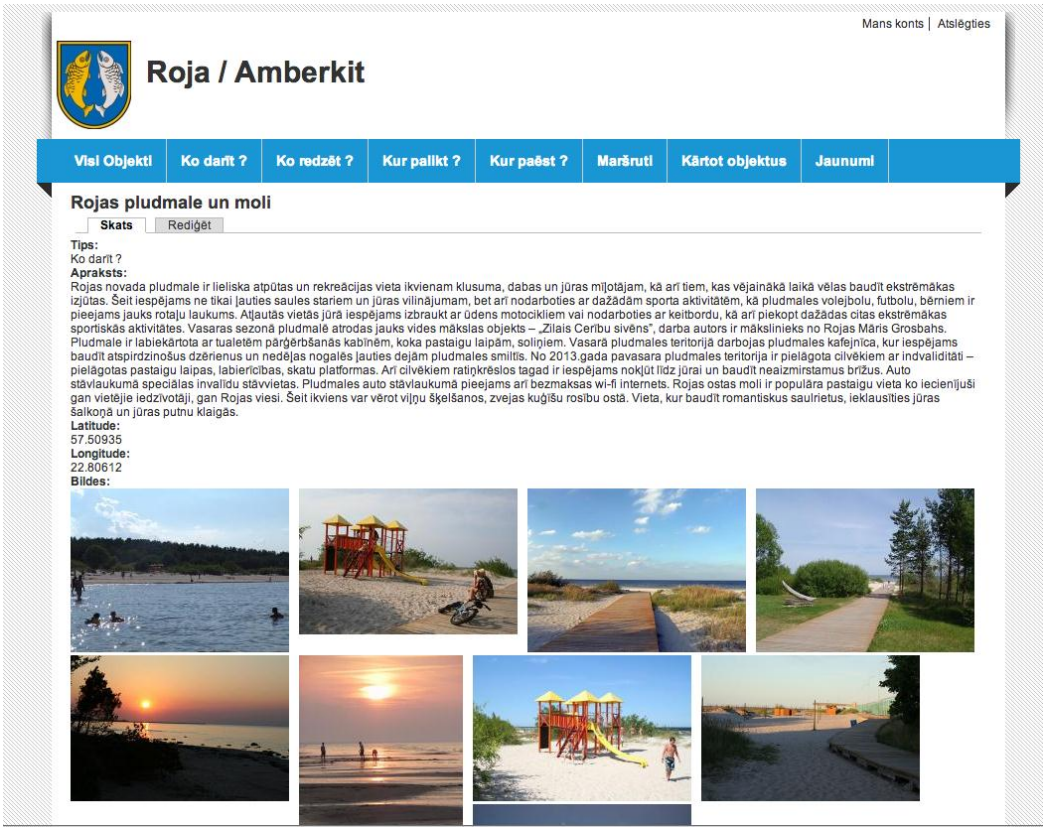
2.1 att. **Rojas AmberKit konta sākuma skats ar *Ko darīt?* Objektiem**



2.2. att. **Jauna objekta pievienošana AmberKit skats**



2.3. att. **Objektu kārtošanas skats AmberKit (kādā kārtībā objekti tiek rādīti lietotnē)**



2.4. att. **Konkrēta objekta skats AmberKit**

2.5. att. **Jaunumu skats AmberKit**

3. pielikums. Lietotnes „Roja Travel Guide” lietošanas instrukcija

Lietotne „Roja Travel Guide” nodrošina informāciju par tūrismu objektiem Rojas novadā. Tūrisma objekti ir sadalīti 5 kategorijās:

- Ko darīt;
- Kur palikt;
- Ko redzēt;
- Kur paēst;
- Maršruti;

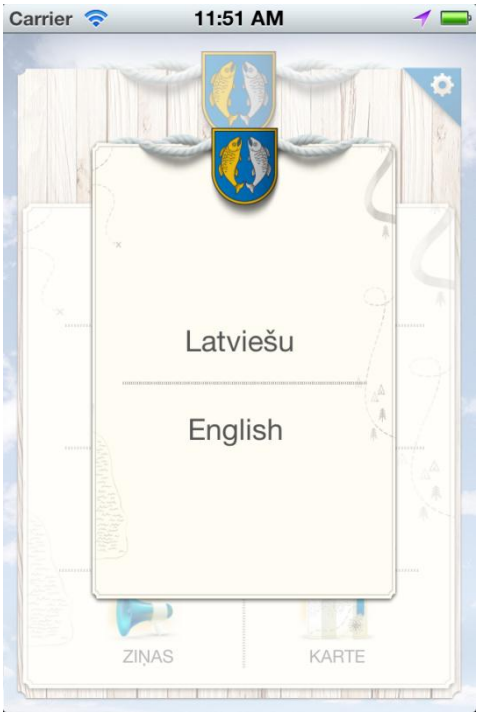
Par katru tūrisma objektu ir iespējams aplūkot detalizētu informāciju, kā arī aplūkot tūrismu objektus kartē. Lietotne piedāvā lietotājiem sekot līdzi novada jaunumiem un ziņām no Rojas novada *twitter* konta.

Atverot lietotni, lietotājam ir iespējams izvēlēties 8 sadaļas:

- Ko darīt;
- Kur palikt;
- Ko redzēt;
- Kur paēst;
- Maršruti;
- Informācija;
- Ziņas;
- Karte.

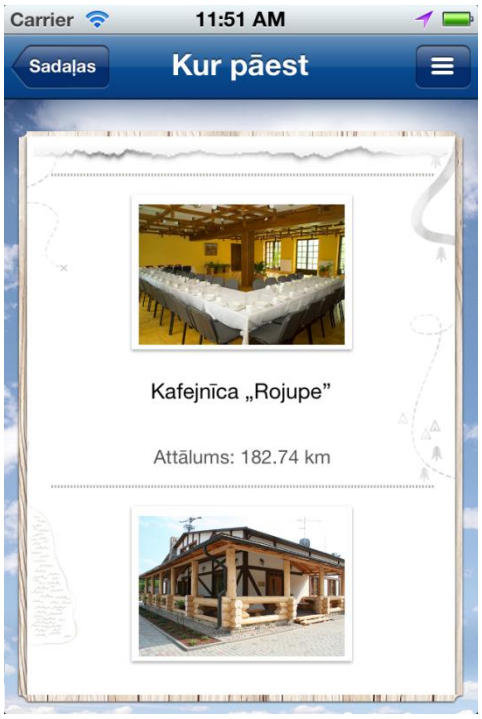
3.1. att. **Rojas novada lietotnes sākuma skats**

Sākuma skatā ir iespējams veikt lietotnes valodas maiņu – latviešu vai angļu.



3.2. att. **Valodas maiņas skats**

Izvēloties kādu no tūrisma sadaļām (Ko darīt, Kur palikt, Ko redzēt, Kur paēst, Maršruti) tiek atvērts skats ar sadaļas objektiem.



3.3. att. **Tūrisma sadaļas skats**

Visos skatos, izņemot sākuma skatu, ir iespējams atvērt sāna izvēlni uz citām sadaļām.

3.4. att. **Sānizvēlnes skats**

Atverot konkrēta objekta skatu, tiek uzrādīti objekta attēli ar iespēju pārslēgt tos, objekta kontaktinformācija, objekta apraksts un iespēja nosūtīt atsauksmi.

3.5. att. **Objekta detalizētais skats**

Izvēloties „Informācija” sadaļu, tiek uzrādīts informatīvs teksts par Rojas novadu.

3.6. att. **„Ziņas” sadaļas skats**

„Ziņas” sadaļā tiek uzrādītas ziņas par Rojas novadu, kā arī iespējams pārslēgties uz Rojas novada *twitter* ziņu skatu.

3.7. att. **„Ziņas” sadaļas skats**

Kartes skatā ir iespējams aplūkot visu tūrismu objektu atrašanās vietu kartē, kā arī iespējams atspējot kādu no kategorijām.

Uzspiežot uz konkrēta objekta, ir iespējams aplūkot to detalizētajā skatā.

3.8. att. **Kartes skats**

Ieteikumus lietotnes funkcionalitātes paplašināšanai sūtiet uz **amberphone@amberphone.lv**

Kvalifikācijas darbs „***Rojas novada tūrisma iOS lietotnes izveide***” 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: ***Armands Baurovskis*** _____ **___.06.2013**.

Rekomendēju darbu aizstāvēšanai

Darba vadītāj**s/a**: ***Dr. dat. Uldis Straujums*** _____ **___.06.2013.**

Recenzents: ***M. dat. Jānis Mārtužs***

Darbs iesniegts 03.06.2013.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Imants Gorbāns* _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2013. prot. Nr. _____

Komisijas sekretārs(–e): _____