LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ANDROID LIETOTNES IZVEIDE ĢIS DATU APSTRĀDEI BEZSAISTES REŽĪMĀ

BAKALAURA DARBS

Autors: **Andris Skudra**
Studenta apliecības nr.: as09632
Darba vadītājs: asociētais profesors Dr.sc.comp. Ģirts Karnītis

RĪGA 2013

ANOTĀCIJA

Mobilo ĢIS lietotņu bezsaistes funkcionalitātes ieviešana šobrīd ir viena no biežāk pieprasītajām funkcijām ĢIS sistēmu veidotājiem. Neskatoties uz lielo pieprasījumu, trūkst pilnvērtīgu ietvaru, kas spētu nodrošināt šo prasību, kā arī esošās iespējas šīs funkcionalitātes ieviešanai ir vāji aprakstītas.

Darbā ir sniegts ieskats ĢIS sistēmās, aprakstīti esošie risinājumi, dažādi mobilo lietotņu veidi, iespējamie kartogrāfijas ietvaru varianti, kas varētu noderēt šādas lietotnes izveidē.

Darbā galvenais uzsvars tiek likts uz izvēlētā ietvara (ArcGIS JavaScript API) risinājuma aprakstiem, pārbaudot tos praksē un veidojot Android lietotni, kas spēj bezsaistes režīmā nodrošināt datu apskatīšanu kartes veidā, kā arī veikt dažādas darbības ar šiem datiem.

Atslēgvārdi: ĢIS, Android, Bezsaistes kartes, HTML5, ArcGIS JavaScript API

ABSTRACT

Mobile GIS application offline functionality is one of the most often requested features by GIS users. Despite the big interest for this, there are not a lot of frameworks that can provide this functionality. Also, existing solutions are not very well described.

This paper contains overview of GIS systems, followed by an existing solution description, comparison between different mobile application types and available frameworks for each application type. Main part of this paper is a list of problem descriptions with possible solutions that can be created using ArcGIS JavaScript API.

Main features of created application are GIS data displaying as map from local sources and map objects querying by geometries and attributes.

Keywords: GIS, Android, Offline Maps, HTML5, ArcGIS JavaScript API

SATURS

TERMINI UN SAĪSINĀJUMI

API – Application Programming Interface, programmēšanas bibliotēka

Android – mobilo tālruņu operētājsistēma

ArcGIS – ESRI piedāvātā ĢIS sistēmas bāze

ArcGIS Android SDK – Android ietvars kartes attēlošanai izmantojot *native* tehnoloģijas

ArcGIS JS API – JavaScript ietvars kartes attēlošanai tīmekļa lietotnēs

ArcGIS REST API (REST interfeiss) - ArcGIS serverim piesaistīts tīmekļa serviss, kas dod piekļuvi ĢIS datiem un funkcijām

CSS – Cascading Style Sheets, tīmekļa dizaina valoda

Datu izgūšana – ĢIS kontekstā apzīmē konkrētas ģeogrāfiskas teritorijas datu atlasi, lai pēc tam šos datus ievietotu, piemēram, mobilajā ierīcē

Ekstents – 4 koordinātas (x,y), kas veido taisnstūri un parasti apzīmē kartes apgabalu, vai arī ģeogrāfiska objekta rāmi

ESRI – firma, kas izstrādā produktu ArcGIS

ĢIS – ģeogrāfiskā informācijas sistēma

HTML – HyperText Markup Language, marķēšanas valoda lapas satura attēlošanai

iOS - mobilo tālruņu operētājsistēma

JS – JavaScript, tīmekļa programmēšanas valoda

JSON – JavaScript Object Notation, JavaScript objekts, kurš pārveidots pārsūtāmā teksta formātā

Kartes slānis – ĢIS kontekstā apzīmē datu (vektor vai rastra) slāni ar ģeogrāfisku informāciju, kas tiek pievienots kartei, lai tiktu attēlots

Kartogrāfijas ietvars (mapping framework) – bibliotēka, kas satur funkcijas kartes zīmēšanai no ĢIS datiem, kā arī papildfunkcijas šo datu apstrādei

Native lietotnes (Native applications) – lietotnes, kas veidotas, izmantojot programmēšanas valodu, kas specifiska platformai, piemēram Android gadījumā Java, iOS gadījumā Objective C

WKID – well known id, telpiskās atsauces identifikators

SDK – Sotfware Development Kit, programmēšanas bibliotēka, kas var ietver arī dažādus izstrādes rīkus

IEVADS

Bezsaistes atbalsts ir īpaši nozīmīgs tieši ĢIS (ģeogrāfisko informācijas sistēmu) nozarē, jo dati ļoti bieži tiek ievākti lauka apstākļos jeb vietās, kur interneta pārklājums nav pieejams – bezsaistes režīmā. Neskatoties uz to, ka organizācija ESRI, kura ir nozares līderis ĢIS izstrādē, piedāvā ietvarus ļoti daudzām ar ĢIS saistītām operācijām, joprojām nav pieejams standartizēts ietvars vai risinājums, kā veikt lielu daļu ar ĢIS saistītām darbībām bezsaistes režīmā, jo īpaši datu sinhronizācijas risinājumi. Šobrīd vienīgais ietvars ar daudz maz pilnīgu bezsaistes funkcionalitāti ir ArcGIS Windows Phone SDK, tomēr šī tehnoloģija ir salīdzinoši novecojusi.

Galvenā pētāmā problēma – trūkst iespējas ĢIS datu apstrādei bezsaistes režīmā, kā arī esošie risinājumi nav pietiekami aprakstīti. Vēlme pēc šādas tehnoloģijas un iespējām ir liela, bet reāli piedāvājums un risinājumi ir maz. Pagaidām ir pieejami atsevišķi risinājumi no ESRI – SDK priekš Android un iOS, bet arī šajos ietvaros atbalsts bezsaistes funkcionalitātei ir salīdzinoši mazs un ietver tikai nelielu daļu no funkcionalitātes, ko pierasts redzēt tiešsaistes ĢIS lietotnēs. Kā arī, šie risinājumi ir ļoti piesaistīti konkrētai ArcGIS servera versijai - 10.1. Praktiski nav aprakstīts standartizēts veids, kā veidot bezsaistes lietotnes ĢIS datu apstrādei. Ir iespējams atrast pāris aptuvenus risinājumus (pārsvarā ideju līmenī), kas nosedz kādu daļu no nepieciešamās lietotnes prasībām.

Pētījuma mērķis - atrast risinājumu biežākajām problēmām ĢIS bezsaistes lietotņu veidošanā un pārbaudīt risinājumu praksē, veidojot Android lietotni. Darbā tika apskatītas tādas problēmas, kā piemēram rastra datu izgūšana, vektordatu izgūšana, dažādas ĢIS datu standarta operācijas (telpiskā atlase, meklēšana pēc atribūtu vērtībām, laukuma aprēķini, telpisko atsauču konvertēšana).

Darba gaitā tika veiktas šādas darbības:

- Ievāktas pamatzināšanas par ĢIS un to datubāzēm, lietotnēm, datu apstrādi, attēlošanu
- Izpētītas mobilo lietotņu izveides iespējas, lietotņu veidi, ietvari
- Identificētas galvenās problēmas (ĢIS kontekstā) risinājuma ieviešanai un lietotnes piemērošanai bezsaistes režīmam:
    - Rastra datu izgūšana, glabāšana un attēlošana
    - Vektordatu izgūšana, glabāšana un attēlošana

    - Vektordatu apstrādes iespējas (atlase pēc atribūtiem, objekta ģeometrijas, laukuma noteikšana)
    - GPS funkcionalitātes pievienošana
    - Ģeometrisko funkciju pieejamība ĢIS kontekstā
- Atrasti esoši risinājumi vai risinājumu idejas bezsaistes funkcionalitātes nodrošināšanai
- Aprakstīts izvēlētais risinājums
- Izveidota lietotne, kas risina augstāk minētās galvenās problēmas

Darba sākumā ir globāli aprakstīta ArcGIS sistēma, tās uzbūves pamatprincipi, galvenās sastāvdaļas un termini. Tālāk ir apskatīti esošie lietotņu piemēri, kas veidoti uz *native* vai tīmekļa tehnoloģijām, un ietver sevī kādu daļu no bezsaistes funkcionalitātes. Pēc tam apskatīti mobilo lietotņu izstrādes pamatveidi, izvērtēti to plusi un mīnusi ĢIS kontekstā un aprakstīta un pamatota izstrādes veida izvēle (starp tīmekļa, *native* un hibrīdlietotni), kā arī izmantotā kartogrāfijas ietvara izvēle (ArcGIS JS API, ArcGIS Android SDK, LeafLet, OpenLayers). Lai gan lietotnes veida izvēle ir ļoti svarīga, šajā pētījumā tai ir sekundāra nozīme. Primārais mērķis ir konkrēta risinājuma apraksts, balstoties uz izdarīto lietotnes veida izvēli. Līdz ar to galvenais uzsvars tika likts uz konkrēta risinājuma aprakstu, nevis vairāku tehnoloģiju salīdzinājumu.

Problēmas iztirzājumā aprakstītas risinājuma galvenās daļas, to apraksts, risinājuma varianti un secinājumi.

Darba rezultātā izveidota Android lietotne, kas izmanto problēmas aprakstā minētos risinājumus un pārbauda to darbību praksē.

1. ĢEOGRĀFISKĀS INFORMĀCIJAS SISTĒMAS

Ģeogrāfiskā informācijas sistēma satur un darbojas ar ģeogrāfiskiem datiem. ĢIS sistēma dod iespēju ievadīt, mainīt un analizēt datus, kuriem ir piesaistīta ģeogrāfiska informācija. ĢIS dati tiek grupēti datubāzes tabulās, un galvenā atšķirība ir tāda, ka katrs objekts var saturēt ģeogrāfisku informāciju, jeb – tā atrašanās vietu.

ĢIS pamats ir geodatubāze, kas tiek būvēta uz parastas (piemēram, Oracle, MSSQL) datubāzes pamata. Geodatubāze spēj glabāt un apstrādāt ģeogrāfisku informāciju. ĢIS programmatūras uzdevums ir nodrošināt sasaisti ar datubāzi un apstrādāt pieprasījumus datiem, kas satur ģeogrāfisku informāciju.

ĢIS galvenais pielietojums ir spēja lietotājam saprotamā formātā attēlot ģeogrāfiskus datus, spēja tos analizēt, filtrēt datus un tos ģeogrāfiski attēlot, piesaistīt objektiem ģeogrāfisku informāciju.

Zemāk sadaļās aprakstītas ArcGIS sistēmas galvenās sastāvdaļas un termini.

1.1. ESRI ArcGIS sistēma

Šajā pētījumā kā bāzes ĢIS tiks izmantots ESRI ArcGIS piedāvātais produkts. ESRI ir kompānija no ASV, kura ir nozares līderis ĢIS sistēmu izveides risinājumu programmatūrā un piedāvā programmatūras komplektu, kas ļauj firmām izveidot savu ĢIS un to paplašināt un pielāgot savām vajadzībām, izmantojot ESRI piedāvātās komponentes.

Tieši šis ĢIS produkts izvēlēts, jo, pirmkārt, piedāvā visplašāko funkcionalitātes klāstu (ĢIS servera funkcijas, datubāzes papildinājums, tīmekļa tehnoloģiju ietvari (Flex, JS, SilverLight), mobilo tehnoloģiju ietvari (Windows Phone, Windows Mobile, Android, iOS), desktop rīki ĢIS datu apstrādei (ArcMap, ArcCatalog) utt. Pēc dažādu informācijas avotu ziņām, ESRI šobrīd pieder 40-70% tirgus savā jomā [1]. Otrkārt, autoram ir pieredze darbā ar šo sistēmu un tehnoloģijām, ko piedāvā ESRI organizācija. Kā arī, autoram ir pieejama šāda ĢIS sistēma, uz kuras datiem tiks veikts šis pētījums.

Galvenais produkts no ESRI produktu kopas ir ArcGIS serveris, uz kura balstās pārējie produkti. Biežāk izmantotās versijas ir 9.3 (Maijs 2009.), 10.0 (Jūnijs 2010.) un šobrīd jaunākā – 10.1 (Jūnijs 2012.). [2] Skatoties mobilo un bezsaistes funkciju kontekstā, 9.3 serveris piedāvāja salīdzinoši minimālu atbalstu, bet līdz ar 10. versijas iznākšanu, ir īpaši piedomāts pie dažādu mobilo iespēju izmantošanas. Tiek piedāvāts

atbalsts vairākām funkcijām, kas svarīgas tieši mobilajām platformām, kā piemēram rastra slāņu eksports uz .tpk formātu, kas ir pielāgots tieši izmantošanai mobilajās platformās, kā arī izmantošanai ArcGIS Android SDK, kurš dod iespējas veidot GIS lietotnes mobilajām platformām.

1.2. ArcGIS serveris

AcrGIS serveris ir viena no centrālajām ArcGIS sistēmas sastāvdaļām. Tas dod iespēju publicēt karšu servisus, kā arī piekļūt tiem, veikt datu analīzi, dažādus ĢIS datu apstrādes uzdevumus, kā arī nodrošina kopējo bāzi sistēmas pilnveidošanai un tās papildināšanai ar jaunām lietotnēm. [3]

ESRI ArcGIS serverī karšu publicēšana notiek caur servisiem. Katram kartes slānim ir jābūt servisā, lai tas būtu pieejams lietotājam, vai arī pieejams programmētājiem ievietošanai tīmekļa lapā caur ArcGIS programmētāju ietvariem, piemēram ArcGIS Javascript API. Servisu direktorija ir pieejama caur REST interfeisu http://<argis-server-url>/arcgis/rest, kur ir apskatāmi visi pieejamie servisi, slāņi, no kā sastāv serviss un informācija par slāņiem, piemēram, koordinātu sistēma, minimālais/maksimālais mērogs u.tml.

1.3. Kartes attēlošana, kartogrāfijas ietvari

Viena no galvenajām ĢIS iezīmēm – karte. Ja reiz sistēma satur ģeogrāfiskus datus, būtu nepieciešams tos attēlot. Kartes attēlošana ir viena no ĢIS sarežģītākajām lietām, tāpēc šis uzdevums lielā mērā tiek uzticēts dažādiem ietvariem un netiek veidots atsevišķs risinājums katru reizi. Šos ietvarus parasti sauc par kartogrāfijas ietvariem (*mapping framework*). Katrs ĢIS pieprasījums uz kartes attēlošanu parasti ir apjomīgāks, nekā standarta tīmekļa pieprasījums, jo atgriežamais datu apjoms ir lielāks, dēļ informācijas apjoma, ko satur standarta kartes skats.

Katra karte sastāv no viena vai vairākiem slāņiem. Slāņi var būt dažāda veida, jo katram no tiem var būt savs uzdevums un datu tipi, tāpēc tiem ir nepieciešama atsevišķa apstrāde. Zemāk aprakstīti divi galvenie slāņu veidi.

1.3.1. Vektordatu slānis (*feature layer, vector layer*)

Vektordatu slāņa objekti satur objekta ģeometriju un atribūtus. Šo slāņu objektu ģeometrijas tiek glabātas vektoru formātā, un sastāv no viena līdz vairākiem punktiem. Grafiskie šie dati tiek atspoguļoti pamatā kā punkti, līnijas un poligoni. Šie

dati tiek glabāti datubāzes tabulā, un no parastas datubāzes tabulas galvenokārt atšķiras ar to, ka satur laukus, kas attiecas uz ģeogrāfisko informāciju, jeb koordinātas.

Vektordatu slāņi ArcGIS terminoloģijā tiek saukti par „Feature Layer". Katrs slānis var saturēt tikai viena ģeogrāfiska tipa objektus – punktus, līnijas vai poligonus [4]. Slāņa objektiem ir arī kopīga telpiskā atsauce, kā arī atribūtu kolonnas.

Datu glabāšana vektoru formātā nodrošina iespēju veikt ar šo slāņu objektiem dažādas ģeogrāfiskās darbības, piemēram:

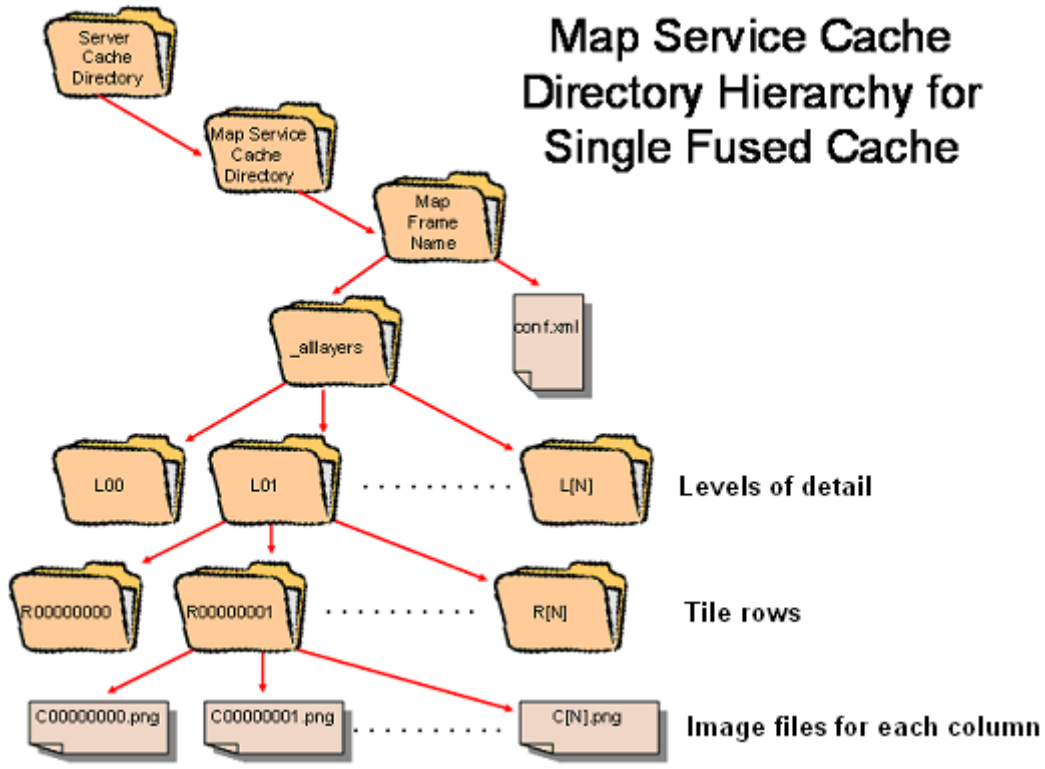
- Izveidot, mainīt objekta ģeometriju
- Atrast objekta platību, garumu
- Noteikt objektu pārklāšanos, krustošanos
- Atrast objektus noteiktā attālumā no kādas koordinātas vai objekta
- Konstruēt buferi objekta ģeometrijai

1.3.2. Rastra datu slānis (*tiled layer, cached layer, raster layer*)

Rastra datu slāņa pamatā ir parasts slānis, kurš ir apstrādāts ļoti ātrai datu atgriešanai. Lai to panāktu, ir nepieciešams izveidot kartes kešdarbes slāni (*map cache, tiled cache*) bilžu formātā, kas satur ģeogrāfiskās informācijas attēlojumu, kurš jau ir izveidots kā bilde. Varētu teikt, ka slānis ir jau iepriekš uzzīmēts, tādējādi nav nepieciešams atsevišķi apstrādāt un zīmēt katru objektu kas atrodas interesējošajā teritorijā, bet ir jāpieprasa konkrētā bilde, kas satur objektus statiskā formātā. Objekti tiek pieprasīti, balstoties pēc 3 parametriem:

- Palielinājuma līmenis jeb zoom
- Rinda
- Kolonna

Kešs tiek izkārtots pa mapēm, parasti struktūra izskatās apmēram šādi (attēls 1):



Attēls 1: Rastra attēlu glabāšanas struktūra [5]

Tādējādi, pieprasot kartē apskatīt konkrētu apgabalu, sistēmai ir jāspēj saprast, kāds palielinājums (mērogs), rinda un kolonna ir nepieciešama un attiecīgi pieprasīt serverim attēlu, kas atbilst šīm koordinātām. Parasti katru kartes skatu veido vairāki attēli. Jo mērogs tuvāks reālajam, jo vairāk bildes tas satur.

Kešdarbes slāņus ieteicams veidot tikai tad, ja to dati reti tiek mainīti. Jo katru reizi, kad dati tiek izmainīti, nepieciešams atjaunot šo slāņu attēlus, kas prasa ļoti daudz laika. Līdz ar to šāda veida slāņi bieži vien tiek saukti par statiskiem, jo to dati (attēli) reti tiek atjaunoti.

1.4. Telpiskā atsauce (*spatial reference*)

Telpiskā atsauce definē sevī kartes projekciju jeb koordināšu sistēmu un transformācijas, kas nepieciešamas pāriešanai uz citām telpiskajām atsaucēm. Koordināšu sistēma definē veidu, kādā tiek aprakstītas punkta koordinātas:

- Ģeocentriskā – izmanto x,y,z punktus koordinātes definēšanai, kā centru izmantojot zemes centru
- Sfēriskā jeb ģeogrāfiskā – izmanto leņķus, kas relatīvi Griničas meridiānam un ekvatoram, parasti apzīmē kā garuma un platuma grādus (Longitude and Latitude). [6]

- Dekarta (Cartesian) – plakanā koordinātu sistēma. Izmanto kartes projekciju un dažādas metodes, lai novērstu aprēķinu kļūdas, kas rodas platības attēlojot plakanā sistēmā [7]. Biežāk izmantotā ir Universal Transverse Mercator (UTM) sistēma. Bieži tiek definētas dažādas lokālās šāda veida sistēmas, jo uz mazākām teritorijām kļūdas apjoms ir mazāk būtisks. Tieši Mercator tipa projekcijas visbiežāk tiek izmantotas tīmekļa karšu lietotnēs.

Lai telpiskās atsauces varētu atšķirt, katrai no tām ir savs identifikators (ID). ĢIS sistēmās šos identifikatorus definē kā SRID, jeb Spatial Reference ID. Identifikators sastāv no 2 daļām:

- Authority - organizācija, kas veido šo daļu no SRID saraksta. Šajā darbā bieži tiks izmantoti šādi Authority:
  - WKID – Well Known ID. ESRI veidots.
  - EPSG – European Petroleum Survey Group.
- Number – skaitlisks identifikators

Bieži vien dažādu organizāciju saraksti satur vienādas telpiskās atsauces, kurām ir dažādi identifikatori katras organizācijas sistēmā. Kā arī, dažādas ĢIS sistēmas vai ietvari mēdz izmanto tikai vienas organizācijas sarakstus. Tādos gadījumos nepieciešams zināt, kā pāriet no vienas organizācijas saraksta uz otru. Šo informāciju iespējams atrast vietnē http://www.spatialreference.org/, kas satur plašu klāstu ar dažādiem SRID dažādās organizāciju sistēmās (pārsvarā WKID un EPSG). [8]

Zemāk norādītas šajā darbā bieži aprakstītās atsauces, kā arī to ID dažādās sistēmās, kurš norādīts iekavās.

- LKS92 (WKID: 3059, EPSG:3059) – Latvijas koordinātu sistēma. Šī darba ietvaros tiks izmantota kā galvenā kartes attēlošanas atsauce - lietotnē, kas tiks veidota pētījuma praktiskajā daļā, visi karšu slāņi (gan rastra, gan vektordatu) izmantos šo atsauci.
- WGS84 Web Mercator (Auxiliary Sphere) - (WKID: 102100, EPSG: 3857) – viena no standarta atsaucēm karšu attēlošanas lietotnēs. Plaši pielietota ESRI sistēmā. Izmanto Dekarta koordinātu sistēmu.
- WGS 84 (WKID: 4326, EPSG:4326) – World Geodetic System. Plaši izmantota GPS pozicionēšanā, izmanto ģeogrāfisko koordinātu sistēmu un koordinātas atgriež garuma un platuma grādos jeb longitude (LON) and latitude (LAT). Šī darba ietvaros tiks izmantota GPS funkcionalitātei, jo HTML5 iebūvētā

lokācijas noteikšana rezultātus atgriež šajā atsaucē. Šī darba ietvaros tiks izmantota arī ģeogrāfisko aprēķinu veikšanai, piemēram ģeogrāfisko objektu garuma un laukuma aprēķiniem, jo pētījumā izmantotais ietvars atbalsta šos aprēķinus tikai izmantojot šo telpisko atsauci.

1.5. Kartes ekstents (*map extent*)

Kartes ekstents norāda kartes apgabalu vai reģionu, ko attēlos karte. Būtībā tas ir rāmis – minimālais un maksimālas garums un platums. Tie ir 4 skaitļi, kas attiecīgi norāda xmin, ymin, xmax un ymax koordinātas. Attēlojot dinamiskas kartes, ekstents mainās, veicot darbības ar karti – mainot mērogu un pozīciju.

1.6. ArcObjects

ArcObjects ir ESRI izstrādāta klašu bibliotēka, kas palīdz veikt darbības ar ģeogrāfiskiem objektiem un veidot dažādas ģeometriskas funkcijas ar datubāzē esošajiem vektordatiem. ArcObjects ir rakstīts C++ valodā, tomēr piedāvā interfeisus uz C# un Java valodām.

ArcObjekti vēl nesen (pirms ArcGIS Server 9.3 versijas) bija vienīgais veids, kā no programmēšanas viedokļa piekļūt ĢIS ģeogrāfiskajiem datiem un tos apstrādāt. Līdz ar ArcGIS Server 9.3 versiju ir parādījies arī otrs variants – servera REST interfeiss, kas dod piekļuvi servera puses funkcijām izmantojot pareizi sastādītu URL. Lai gan REST interfeiss ir ērtāks un ātrāks, ArcObjekts piedāvā plašāku funkcionalitāti un joprojām nav pilnībā aizvietojams ar REST interfeisa piedāvātajām funkcijām.

1.7. ArcGIS REST API jeb REST interfeiss

REST interfeiss piedāvā piekļuvi ĢIS datiem caur URL, izmantojot interfeisā piedāvātās funkcijas. REST interfeisa bāzes adrese ir http://<argis-server-url>/arcgis/rest, kurā redzami visi pieejamie servisi. Tālāk var apskatīt katru servisu atsevišķi, redzēt tam piederošos datu slāņus kā arī citus datus, piemēram ekstentu, telpisko atsauci, servisa izveidotāju u.tml.

Ir pieejams grafisks interfeiss, kur funkciju parametrus iespējams aizpildīt HTML formā, tomēr biežāk šīs funkcijas tiek izsauktas manuāli sastādot URL, jo īpaši ĢIS tīmekļa lietotnēs. Sākotnēji interfeiss atbalstīja tikai datu apskatīšanu, bet

līdz ar 10. versiju ir pieejama arī datu labošana. REST interfeiss pārsvarā tiek izmantots ĢIS tīmekļa lietotnēs, veicot datu pieprasījumus ĢIS datubāzei.

Biežāk izmantotās REST interfeisa funkcijas ir:

- Query – atgriež vaicājuma datus, var atlasīt pēc ģeogrāfiskām īpašībām vai arī lauku vērtībām
- Export – eksportē kartes skatu, norādot, piemēram, koordinātas, formātu, slāņu sarakstu
- Geometry Service dažādas funkcijas, piemēram:
    - Lenght (līnijas, poligona garums),
    - Area (poligona laukums),
    - Buffer (ģenerē bufer-zonu punktam, līnijai vai poligonam un atgriež šo ģeometriju)
    - Simplify – ģeometrijas vienkāršošana (samazina punktu skaitu, aizvācot blakus esošus, salīdzinoši tuvus punktus)

2. PROBLĒMAS AR BEZSAISTES FUNKCIONALITĀTES IEVIEŠANU

Neskatoties uz to, ka ĢIS pārsvarā tiek izmantots lauka apstākļos, bezsaistes režīms ĢIS lietotnēm vēl ne tuvu tik pieejams un attīstīts, kā tam vajadzētu būt. Kā arī, par tā izmantošanu un pieejamajām funkcijām ir ļoti maz informācijas. Ļoti bieži ĢIS datu apskate un ievākšana notiek apstākļos, kur internets nav pieejams vai ir pārāk lēns, lai pilnvērtīgi un ātri saņemtu apjomīgos ĢIS datus. Lietotņu ietvari pārsvarā ir koncentrēti uz datu iegūšanu un attēlošanu, izmantojot interneta pieslēgumu un publiskos resursus internetā.

Piemēram, tāda ĢIS standarta funkcionalitāte kā ģeogrāfiskie vaicājumi ir izpildāmi tikai servera pusē, un bezsaistes režīmā šīs problēmas apiešanai tiek izmantotas diezgan dažādas metodes. Klienta puses ietvaros (piemēram, ArcGIS Javascript API) vispār nav pieejama daļa standarta ģeogrāfisko funkciju, vai arī ir pieejamas tikai daļēji, nemaz nerunājot par atbalstu datu izgūšanai.

Kartogrāfijas ietvaru (ArcGIS JS API, ArcGIS Android API, LeafLet utt) aprakstos ir plašs klāsts ar funkcijām, tomēr daudzas no tām fonā izmanto pieprasījumus uz serveri, kā rezultātā bezsaistes režīmā nav izmantojamas. Trūkst pārskatāma apraksta ar tieši bezsaistes režīmā pieejamajām funkcijām, kas neizmanto servera puses funkcijas.

Vēl viena standarta problēma bezsaistes lietotnēm ir datu izgūšana un sinhronizācija uz serveri. Runājot par ĢIS datu sinhronizāciju, šobrīd vienīgais esošais risinājums ir pieejams ESRI Windows Mobile platformai, tomēr šis risinājums ir lēns un nestabils. Tā kā sinhronizācijas laikā nemitīgi notiek komunikācija starp datoram pieslēgtu Windows Mobile ierīci, pašu datoru un serveri, tas rada papildus problēmas un nestabilitāti, jo procesā pārāk daudz ir iesaistīta ierīce.

Sinhronizācija ir viens no sarežģītākajiem moduļiem šādās lietotnēs. Situāciju neatvieglo arī ESRI izvēlētā sinhronizācijas stratēģija, kura īpašu uzmanību pievērš datu integritātei. Standarta sinhronizācijas process ir šāds:

- Tiek salīdzinātas datu shēmas,
    - ja tās atšķiras, sinhronizācija tiek pārtraukta.
    - ja tās sakrīt, dati tiek sūtīti uz serveri un ievietoti datubāzē

- Pēc datu nosūtīšanas, tos atkal ielādē ierīcē, un salīdzina ar vecajiem datiem
- Ja ielādētie dati atšķiras no vecajiem datiem, tiek uzskatīts, ka sinhronizācijas laikā ir radusies kļūda un ne visi dati ir nonākuši līdz datubāzei, un tiek atceltas visas veiktās izmaiņas.

Lai gan šāds process rada papildus drošību, ka visi dati nonāks datubāzē, tomēr tas ir pārāk lēns. Galvenokārt tāpēc, ka pārāk daudz darbības šajā procesā tiek veiktas uz ierīces, kurai tomēr ir krietni zemāka ātrdarbība. Tāpēc, iespējams, sinhronizācijas un datu izgūšanas daļu būtu ieteicams veidot kā atsevišķu lietotni vai moduli uz servera puses tehnoloģijām, kas izmanto ierīces savāktos datus. Tādā veidā tiek izlaists posms, kurā mobilajai ierīcei nepieciešams sazināties ar serveri, līdz ar to tiek paātrināts šis process, kā arī mazinās tā sarežģītība.

3. ESOŠĀS LIETOTNES UN RISINĀJUMI, KO IESPĒJAMS ATRAST INTERNETĀ

Sadaļā aprakstīti internetā atrodamie projekti, kas paredzēti lietošanai uz Android platformas bezsaistes režīmā. Lielākā daļa no risinājumiem nesatur vektordatu slāņu izmantošanu, izņemot WebMapSolutions risinājumu, bet šim risinājumam kods diemžēl nav pieejams. Visi zemāk minētie risinājumi tikai daļēji satur funkcionalitāti, kas būtu nepieciešama pilnvērtīgai ĢIS bezsaistes lietotnei.

Risinājumi pārsvarā ir balstīti uz tīmekļa tehnoloģijām, kas dod iespēju tos izmantot uz jebkuras platformas, tai skaitā uz mobilajām platformām. Divi no šiem risinājumiem izmanto PhoneGap hibrīdlietotņu ietvaru, kas nodrošina to, ka iespējams piekļūt tādām ierīces funkcijām, kas nebūtu pielietojamas, izmantojot tikai tīmekļa tehnoloģijas.

3.1. WebMapSolutions risinājums

Lietotnes apraksts - http://www.webmapsolutions.com/mobile-arcgis-online-feature-layers

Līdz šim vispilnīgākā lietotne, kādu iespējams atrast izmantošanai bezsaistes režīmā. Risinājumā tiek izmantoti ArcGIS Flex, ArcGIS JS API un LeafLet kartogrāfijas ietvari. Lietotne ietver ne tikai bezsaistes režīmu, bet spēj arī darboties ar aktuālajiem datiem tiešsaistes režīmā. Šos režīmus iespējams dinamiski pārslēgt. Pagaidām vienīgā lietotne, kas nodrošina iespēju bezsaistē veikt vektordatu labošanu, kā arī šos datus pēc tam nosūtīt uz serveri un saglabāt.

Rastra dati lokāli uz ierīces tiek glabāti .tpk failā, ko iespējams izgūt caur ArcGIS 10.1 serveri, norādot interesējošo apgabalu, piemēram, pēc objekta ģeometrijas. Savukārt vektordati tiek glabāti JSON formātā.

3.2. Android SDK Windturbine inspector

Lietotnes apraksts - http://www.arcgis.com/home/item.html?id=01acc4eef0f04c2d9d0796e78938d3cd

Lietotne veidota, izmantojot ArcGIS Android SDK 10 versiju (2012. gada jūnijs) kā piemērs esošajām bezsaistes iespējām. Vektordati tiek glabāti JSON formātā, izmantojot *JSON Feature Sets* slāņu klasi. Savā ziņā šie slāņi izmanto līdzīgu

formātu, kādu iespējams izgūt no ArcGIS REST interfeisa Query funkcijas (skatīt 1.7 sadaļu).

Rastra dati tiek glabāti .tpk failā, līdzīgi kā WebMapSolutions risinājumā.

Lietotne nodrošina datu izgūšanu un lietošanu bezsaistes režīmā, kā arī jaunu ģeogrāfisko punktu pievienošanu caur GPS ierīci. Galvenais trūkums - līniju un poligonu pievienošana šajā lietotnē nav pieejama. Kā arī, datu automātiska sinhronizācija atpakaļ uz serveri nav pieejama. Balsoties uz esošo aprakstu, jāveido atsevišķs modulis, lai uzkrāto informāciju varētu nogādāt uz serveri.

3.3. LeafLet + TileMill lietotne

Lietotnes apraksts - http://geospatialscott.blogspot.com/2012/04/phonegap-leaflet-tilemill-offline.html

Lietotne veidota kā piemērs rastra datu izmantošanai bezsaistes režīmā. TileMill ietvars ļauj izveidot rastra datu kešdarbes slāni .mbtiles formātā (praktiski mobilā datubāzē sqlite). TileMill izmantošanai ir nepieciešams PhoneGap ietvars (nodrošina piekļuvi pie mobilās datubāzes), kā arī tas nodrošina iespēju lietotni izmantot jebkurai platformai, ko atbalsta PhoneGap ietvars.

Savukārt LeafLet ietvars ir domāts karšu attēlošanai un izmanto šo sagatavoto .mbtiles formāta kešdarbes slāni jeb rastra slāni, lai lietotājam attēlotu karti, kas sastāv no šiem iepriekš sagatavotajiem attēliem. LeafLet ir viens no populārākajiem ietvariem karšu attēlošanas lietotnēm, un tas ir arī viens no retajiem, kurš atbalsta lokālo datu izmantošanu kartes attēlošanai.

Galvenā problēma - risinājums nesatur vektordatu slāņus, un ir izmantojams tikai fona slāņu apskatei.

3.4. HTML5 tiles in local strorage

Lietotnes apraksts - http://help.arcgis.com/en/webapi/javascript/arcgis/jssamples/exp_webstorage.html

Lietotne veidota kā piemērs vienai funkcijai, kas izmanto nodrošina rastra datu lokālu saglabāšanu, izmantojot interneta karšu servisus. Viens no retajiem bezsaistes funkcionalitātes piemēram, kas atrodams starp ESRI JS piemēriem. Izmanto jauno HTML5 tehnoloģiju local storage, kas ļauj glabāt objektus pārlūkā, izmantojot key/value pārus. Piemērs izmanto tīmekļa karšu servisu, kuram tiek pieprasīts attēls. Ja šis attēls vēl nav saglabāts lokāli, lietotne to saglabā un nākošreiz, kad tas

nepieciešams ielādē pārlūkā saglabāto attēla versiju, nevis veic pieprasījumu uz karšu servisu.

Galvenā problēma ir local storage atmiņas ierobežojums – 5MB. Rastra datiem nepieciešams krietni vairāk atmiņas.

4. LIETOTNES VEIDA IZVĒLE

Sadaļā aprakstītas galvenās izvēles iespējas, dažādi aspekti, kas jāņem vērā, izvēloties, ar kādām tehnoloģijām tiks izstrādāta ĢIS lietotne bezsaistes datu apstrādei. Aprakstīti plusi un mīnusi globālā kontekstā, kā arī ĢIS kontekstā. Aprakstīts izmantojamo tehnoloģiju apraksts, iespējas un trūkumi. Runājot par mobilo lietotņu veidiem, vienmēr ir pieejami 3 varianti:

- Tīmekļa lietotne
- Hibrīdlietotne
- *Native* lietotne - lietotne, kas veidota, izmantojot programmēšanas valodu, kas specifiska platformai, piemēram, Android gadījumā Java

Katram no veidiem ir savi plusi un mīnusi, bet tieši ĢIS lietotnes gadījumā veida izvēle balstīties uz katram veidam pieejamajiem kartogrāfijas ietvariem un iespējām, ko tie piedāvā tieši sakarā ar bezsaistes ĢIS funkcionalitāti. Kartogrāfijas ietvara galvenā funkcija ir spēja attēlot ģeogrāfiskos datus kā karti. Līdz ar to galvenais ir izvēlēties kādu no šiem ietvariem, nevis pašu lietotnes veidu. Ietvara izmantošana šāda veida lietotnēs ir praktiski obligāta.

Mobilās lietotnes veida un izmantotā kartogrāfijas ietvara izvēle ir viens no pirmajiem un būtiskākajiem soļiem kas jāveic, uzsākot veidot mobilo lietotni, jo vairāki turpmākie risinājumi un problēmas izrietēs no šīs izvēles. Tāpēc ir nepieciešams apskatīt vairākus aspektus, uz kuriem balstoties būtu iespējams izvēlēties optimālāko risinājumu. Tā kā mobilā lietotne veic darbības ar ĢIS datiem, izvēlē ir iesaistīti vairāki papildus nosacījumi, kuri tiks aprakstīti zemāk. Galvenais problēmu radītājs šajā situācijā ir bezsaistes iespēju nepieciešamība un ĢIS datu apstrāde.

Pats pirmais un svarīgākais nosacījums izvēlē – vai izmantojot dotās tehnoloģijas, kas pieejamas konkrētajam veidam vispār iespējams attēlot karti bezsaistes režīmā – respektīvi, vai ir iespējams attēlot karti, kura attēlojamos slāņus nolasa no lokāliem failiem, kas saglabāti ierīces atmiņā. Šis punkts sevī ietver arī datu izgūšanu un novietošanu lokāli uz mobilās ierīces.

Otrs svarīgākais nosacījums būtu spēja pielietot dažādas standarta ģeogrāfiskās funkcijas - telpiskā atlase, meklēšana pēc atribūtu vērtībām, laukuma aprēķini, telpisko atsauču konvertēšana, GPS.

Trešais nosacījums būtu iespēja veidot lietotni maksimāli elastīgu, nepiesaistītu kādai konkrētai ArcGIS versijai, kā arī priekšrocība būtu, ja risinājums būtu viegli pielāgojams darbībai dažādās mobilajās platformās. Iespējams pat izmantojams pavisam citās (ne ArcGIS) sistēmās.

Zemāk aprakstīti visi 3 mobilo lietotņu veidi, kā arī tiem pieejamie kartogrāfijas ietvari.

4.1. Tīmekļa lietotne

Tīmekļa lietotnes balstās uz tīmekļa tehnoloģijām (HTML, CSS, JS) un tās tiek lietotas, izmantojot mobilo pārlūku. Galvenais pluss ir tas, ka lietotne nav atkarīga no platformas un viens risinājums der visām platformām. Vienīgais iemesls iespējamajām saderības problēmām varētu būt pārlūku dažādais atbalsta līmenis HTML5 jaunākajām funkcijām, kā arī dažādu mobilo platformu pārlūku atšķirības. Tā kā izstrādātāji ir vairāk pazīstami ar tīmekļa tehnoloģijām nekā *native*, tad izstrādes ērtums un patērētais laiks lielākajai daļai izstrādātāju varētu likt nosvērties par labu šim risinājumam.

Viena no problēmām varētu būt lietotāja interfeisa izveide. Tīmekļa tehnoloģijām pieejamās saskarnes veidošanas iespējas vēl nav gluži tādā līmenī, lai varētu simulēt ierīces saskarni tā, lai tā izskatītos kā to pierasts redzēt *native* lietotnēm. Lai problēmu mazinātu, ir pieejami vairāki JS ietvari, kas paredzēti tieši mobilajām platformām, kā dažus var minēt Dojo Mobile, jQuery Mobile, ResponsiveJS.

Otra būtiska problēma ir nespēja piekļūt ierīces iespējām, piemēram, kontaktu sarakstam, kamerai, mikrofonam. Tomēr, pateicoties HTML5, jau šobrīd ir pieejamas vairākas citas svarīgas funkcijas no ierīces piedāvātajām – GPS, akselerometrs, failu augšupielāde [9].

4.1.1. Bezsaistes tīmekļa lietotne

Tā kā lietotne tiek veidota bezsaistes režīmam, nepieciešams apskatīt arī šīs iespējas pieejamību tīmekļa lietotnēm. Bezsaistes tīmekļa lietotnes nosacīti var iedalīt 2 grupās - daļēji bezsaistes un pilnībā bezsaistes.

Daļēji bezsaistes lietotnes domātas pārsvarā izmantošanai caur internetu, bet spēj daļēji vai pilnībā nodrošināt darbību arī bezsaistes režīmā. Šī funkcionalitāte balstās uz HTML5 bezsaistes iespējām – *cache manifest* failiem [10]. Šajos failos tiek

norādīti lietotnei piederošie faili, kurus nepieciešams lokāli saglabāt un atjaunot tikai tad, ja failos ir notikušas izmaiņas. Līdz ar to, lietotāja pārlūka kešatmiņā vienmēr tiek glabātas failu kopijas, kas tiks izmantotas gadījumā, ja no tiešsaistes režīma ierīce pēkšņi nonāk bezsaistes režīmā.

Pilnībā bezsaistes lietotnes nesatur servera puses funkcionalitāti, viss kods tiek veidots tikai uz klienta puses tīmekļa tehnoloģijām. Visi pirmkoda faili tiek novietoti uz lietotāja ierīces, līdz ar citiem datiem, kas nepieciešami ierīces darbībai. Šīs lietotnes praktiski vienmēr tiks veidotas kā vienas lapas lietotnes (*single-page applications*), norādes uz CSS un JS failiem tiek ievietotas, izmantojot relatīvās saites, kas nodrošina to, ka lietotni iespējams ērti pārvietot starp dažādām lokācijām un izplatīt.

4.1.2. Tīmekļa lietotne ĢIS kontekstā

Veidojot tīmekļa ĢIS lietotni, būtu jāizmanto kāds no JavaScript kartogrāfijas ietvariem. Ņemot vērā, ka šajā darbā aprakstītā lietotne tiks veidota darbam ar ArcGIS sistēmu, pirmais variants būtu izmantot ArcGIS veidoto JS ietvaru, jeb pilnā nosaukumā ArcGIS JS API. Citas alternatīvas būtu:

- LeafLet ietvars – viens no populārākajiem kartogrāfijas ietvariem, labs atbalsts lokālo rastra slāņu izmantošanā kombinācijā ar MbTiles ietvaru, kurš paredzēts rastra datu apstrādei.
- OpenLayers – arī salīdzinoši populārs, labs atbalsts tieši lokālo vektordatu slāņu izmantošanā (GeoJSON, KML un citos formātos)

Viens no galvenajiem JS kartogrāfijas ietvaru trūkumiem ir ĢIS datu analīzes funkciju trūkums. Piemēram, objektu šķelšanās teritorijas laukuma noteikšana, dažādas koordinātu sistēmu konvertācijas funkcijas. Šīs funkcijas vienmēr ir bijušas servera pusē, un JS ietvaros praktiski nav sastopamas. Līdz ar to, bieži vien bezsaistes lietotnēs šādas funkcijas nemaz nav pieejamas.

Otrs trūkums ir datu izgūšanas un sinhronizācijas iespēju trūkums. Tīmekļa lietotņu kontekstā šo funkcionalitāti visticamāk būtu nepieciešams veidot kā atsevišķu lietotni vai moduli, kas darbojas ar servera pusē.

4.2. *Native* lietotne

Native lietotnes tiek būvētas katrai mobilajai platformai atsevišķi, līdz ar to jāizmanto valoda, kas nepieciešama konkrētajai platformai, Android gadījumā Java.

Atšķirībā no tīmekļa lietotnēm, *native* lietotnes piedāvā augstāku veiktspēju, kā arī pieeju visām ierīces iespējām. Izplatīšana notiek caur lietotņu „veikaliem" – Android gadījumā tas ir Google Play. Šādas lietotnes prasa lielas izveidošanas un uzturēšanas izmaksas, bieži vien izstrādātājiem nākas apgūt jaunas tehnoloģijas un ierīces īpatnības. Līdz ar to, autoraprāt, *native* izstrādes tehnoloģijas būtu ieteicams izmantot tikai tad, ja lietotne prasa augstu ātrdarbību, kā arī *native* interfeisa nepieciešamība ir obligāta.

4.2.1. *Native* lietotne ĢIS kontekstā

ĢIS lietotnes veidošana uz *native* tehnoloģijas nozīmētu ArcGIS Android SDK kartogrāfijas ietvara izmantošanu, kas ir praktiski vienīgais šobrīd pieejamais Java ietvars uz ĢIS lietotņu veidošanai. Šobrīd jaunākās ir 2.0 versija (jūnijs 2012) un 10.1.1 (janvāris 2013) un tas atbalsta 95% Android ierīču – sākot ar 2.2 Android versiju [9].

Ja neskaita vispārīgās *native* izstrādes modeļa problēmas, tad citu ierobežojumu šajā gadījumā nav.

Galvenais ArcGIS Android SDK pluss tas, ka tas bezsaistes režīmā atbalsta krietni vairāk funkciju, nekā JS API. Piemēram, kā rastra datus tas spēj izmantot gan *compact cache* (.tpk fails, iegūstams ar 10.1 servera REST export funkciju), gan *tile cache* (rastra dati, savietoti ArcGIS Server cache formātā, respektīvi, mapju struktūra, skat. 1.3.2 sadaļu). Savukārt lokālie vektordatu slāņi tiek glabāti un attēloti līdzīgi kā JS API – izmantojot JSON failus. Ir arī labāks atbalsts dažādām ģeometriskajām funkcijām – laukuma aprēķiniem, šķelšanās noteikšanai, objektu identificēšanai utt. Tas ir galvenokārt tāpēc, ka Java vidē ir daudz vieglāk simulēt servera puses funkcijas, jo ArcGIS Android SDK tiek izmantoti moduļi no ArcGIS Server Java versijas.

Šobrīd ArcGIS Android SDK jaunākā versija 10.1.1 satur praktiski tikai datu apskates iespējas bezsaistes režīmā. Tomēr drīzumā ESRI sola jaunu versiju, kura piedāvās arī datu labošanas un sinhronizācijas funkcionalitāti.

4.3. Hibrīdlietotne

Hibrīdlietotne ir vidusceļš abām iepriekšējām izstrādes tehnoloģijām. To veido, izmantojot tīmekļa tehnoloģijas, bet komplektē kā *native* lietotni, kas izmanto WebView komponenti, kas ļauj lietotnes logā izpildīt ar tīmekļa tehnoloģijām rakstītu

kodu [9]. Hibrīdlietotņu veidošanā krietni palīdz dažādi izveidotie ietvari, piemēram PhoneGap, Sencha Touch vai Appcelerator Titanium. Izplatīšana notiek caur lietotņu veikaliem, tāpat kā *native* lietotnēm.

Galvenais pluss ir iespēja salīdzinoši viegli šo lietotņu pirmkodu piemērot citām platformām. Kā arī, lietotnes atbalsta vairāk ierīces funkciju, nekā tīmekļa lietotnes.

4.3.1. Hibrīdlietotne ĢIS kontekstā

Hibrīdlietotnes veidošanas gadījumā būtu jāizmanto kāds no iepriekš minētajiem JavaScript kartogrāfijas ietvariem (skatīt 4.1.2), tāpat kā tīmekļa lietotnei. Kā arī, būtu jāizmanto kāds no augstāk minētajiem hibrīdlietotņu ietvariem. Tā kā ĢIS funkciju izmantošanai nepieciešamā geolokācijas funkciju ir pieejama arī caur HTML5, ĢIS kontekstā hibrīdlietotnes veidošana nekādu papildus funkcionalitāti (salīdzinot ar tīmekļa lietotni) nespēj sniegt.

4.4. Salīdzinājums, izvēle un pamatojums

Balstoties uz sadaļā aprakstīto informāciju, beigās kā divi potenciāli šim uzdevumam piemērotākie ietvari tika izvēlēti ArcGIS JS API (veidojot bezsaistes tīmekļa lietotni, pēc vajadzības izstrādei pievienojot PhoneGap ietvaru, ja tomēr būs nepieciešama kāda papildus funkcionalitāte no ierīces), un ArcGIS Android SDK (veidojot *native* bezsaistes lietotni). Zemāk aprakstīts abu ietvaru salīdzinājums pēc ievadā minētajiem kritērijiem.

Salīdzinot kartes attēlošanas iespējas jāsecina, ka *native* lietotnēm, kas balstītas uz ArcGIS Android SDK, šis process ir krietni vienkāršāks. Ir pieejams skaidrs apraksts, kā pieslēgt kartei gan vektor, gan rastra slāņus, kamēr ArcGIS JS API ir rūpīgi jāmeklē risinājums, to papildinot savām vajadzībām. Tomēr to var atrast (aprakstīts 5.4 nodaļā).

Dažādu ģeogrāfisko darbību funkcijas ir pieejamas bezsaistes režīmā gan ArcGIS JS API, gan ArcGIS Android SDK, tomēr Android risinājums ietver sevī vairāk funkciju, jo ir vairāk pielāgots tieši bezsaistes režīmam.

Risinājuma elastīgums ir sadaļa, kurā ArcGIS Android SDK ir stipri piesaistīts tieši ArcGIS sistēmai, konkrēti 10.1 versijai. Tas ir tāpēc, ka šī ietvara pirmā versija tika izlaista sākot tieši ar servera 10.1 versiju, un pirms tam netika īpaši domāts ne par viedtālruņu ĢIS lietotnēm, ne par bezsaistes iespējām. Līdz ar to ArcGIS Android SDK izmantošana nozīmētu, ka ir obligāti nepieciešama salīdzinoši jauna ArcGIS

servera versija. Savukārt JS API izmantošana nozīmētu, ka veidotais risinājums, lai gan pielāgots konkrēta uzņēmuma ĢIS sistēmai, tomēr salīdzinoši neilgā laikā to būtu iespējams pielāgot arī citām sistēmām.

Galvenie izvēles iemesli ir šādi:

- Pirms izvēles tika noskaidrots (praktiskā ceļā), vai ArcGIS API pietiekoši labi atbalsta lokālo rastra un vektorslāņu pievienošanu un izmantošanu kartē. Pieejamā funkcionalitāte bija pietiekama.
- Lai gan ArcGIS Android SDK izmantošana ļoti paātrinātu un atvieglotu tieši izstrādes sākuma posmu (un jo īpaši ĢIS funkcionalitātes ieviešanu), beigās sarežģītāku problēmu (kas nav saistītas tieši ar ĢIS funkcionalitāti) risināšanai parocīgāks būtu tieši ArcGIS JS ietvars un tīmekļa tehnoloģijas.
- Lai gan ArcGIS Android SDK ietvars ir vairāk paredzēts tieši bezsaistes lietošanai nekā JS ietvars, Android SDK tomēr ir ļoti svaigs produkts un nav sevi vēl īsti pierādījis.
- Iespējams daļu funkcionalitātes pārnest no citām sistēmām, kas izmanto tīmekļa tehnoloģijas. Ja reiz tiek veidota mobilā lietotne, vairumā gadījumu ļoti iespējams, ka pirms tās ir uzņēmumā bijusi veidota galddatoru tīmekļa lietotne, līdz ar to iespējams izmantot idejas un funkcionalitāti no šīs lietotnes.
- Autora personīgais iemesls – lielāka pieredze darbā ar tīmekļa tehnoloģijām nekā Java programmēšanas valodu. Līdz ar to tīmekļa tehnoloģiju lietošana varētu paātrināt darba procesu.

Balstoties uz augstāk rakstīto, tika pieņemts lēmums veidot lietotni, izmantojot ArcGIS JS API ietvaru. Sākumā lietotne tiks veidota kā tīmekļa lietotne, un nepieciešamības gadījumā risinājumam tiks pievienots kāds no hibrīdlietotņu veidošanas ietvariem, ja būs nepieciešama šī papildus funkcionalitāte.

4.5. Secinājumi

Šajā jautājumā noteikti nav pareizas vai nepareizas izvēles. Izvēlē lielā mērā balstās uz izstrādātāja personīgajām prasmēm (tīmekļa tehnoloģijas pret *native*), kā arī prasībām pret lietotni – vai svarīgākais ir veiktspēja, funkcionalitātes daudzums (ņemot vērā mobilo tehnoloģiju ierobežojumus, ne vienmēr ir ieteicams censties ieviest pēc iespējas vairāk funkcionalitātes), lietojamība vairākās platformās.

5. GALVENIE ASPEKTI, PROBLĒMAS UN TO RISINĀJUMU VARIANTI ĢIS LIETOTNEI BEZSAISTES REŽĪMAM

Šajā sadaļā tiks aprakstītas galvenās problēmas, kuras rodas, mēģinot ĢIS lietotnes funkcionalitāti izmantot bezsaistes režīmā. **Problēmu risinājumi tiks veidoti, balstoties uz iepriekš veikto izvēli, ka tiks izmantots ArcGIS JS API kartogrāfijas ietvars. Savukārt datu izgūšanas jautājums tiks risināts servera pusē, izmantojot ASP.NET un C#.** Pamatojoties uz 2. nodaļā aprakstītajām problēmām sakarā ar mobilās ierīces iesaistīšanu datu izgūšanas un sinhronizācijas jautājumos, autoraprāt labākais risinājums būtu datu izgūšanu risināt servera pusē un ierīci iesaistīt tikai tajā brīdī, kad jau ir pieejams datu komplekts, ko nepieciešams pārkopēt uz ierīci.

Sadaļā aprakstītās lietas tiešsaistes režīmā pieder pie ĢIS lietotnes pamat funkcionalitātes, tomēr bezsaistes režīmā šo funkciju nodrošināšana rada problēmas vairāku iemeslu dēļ:

- Tās nav paredzētas izmantošanai bezsaistes režīmā
- Internetā trūkst informācijas problēmu novēršanai
- Ātrdarbības problēmas

5.1. Rastra slāņu izgūšana konkrētai teritorijai

Lauka apstākļos lietotājs parasti vēlas strādāt ar datiem, kas pieder tikai kādam konkrētam apgabalam. Lielāks datu apjoms parasti nav nepieciešams, kā arī nemaz nav iespējams – dati aizņem pārāk daudz vietas (piemēram, Latvijas teritorijas ortofoto slānis 11 mēroga līmeņos aizņem pāris gigabaitus), lai tiktu lokāli novietoti uz mobilās ierīces. Ierasts darba scenārijs šādu lietotņu lietotājiem – tiek izgūti dati priekš konkrētas teritorijas, lietotājs dodas uz šo teritoriju, izmanto lietotni ar šiem datiem, iespējams veic izmaiņas un tā tālāk. Vēlāk datus sinhronizē atpakaļ uz serveri. Tā iespējams izgūt datus vairākiem apgabaliem un darboties atsevišķi ar katru no tiem.

Līdz ar to ir nepieciešams veids, kā atfiltrēt rastra datus priekš tā, jeb – zinot nepieciešamā datu apgabala koordinātas, nepieciešams veids kā:

- noteikt, kuri rastra datu direktorijas attēli atbilst šim apgabalam
- nogādāt šos attēlus citā direktorijā, lai lietotājs varētu tos pārkopēt uz ierīci

ArcGIS 10.1. serveris satur funkciju rastra failu eksportam, kā parametru norādot kādu objektu no vektordatu slāņa [11]. Diemžēl ArcGIS 9.3 serverī šāda funkcija nav pieejama, un ir jāmeklē citas iespējas, ko varētu izmantot, lai pietiekami elastīgi nodrošinātu šo funkcionalitāti.

5.1.1. Risinājums

Balstoties uz zināšanām par ArcGIS rastra datu izvietojuma struktūru (skatīt 1.3.2 sadaļu), jānoskaidro kā noteikt nepieciešamo rastra bilžu komplektu konkrētas teritorijas datiem. Izmantojot ESRI sniegto skaidrojumu [12], rindas un kolonnas atrašanās vietu pēc punkta kartē var noteikt pēc šādām formulām:

```
Column = Floor((Point of interest X – Tile origin X) / Ground width
of a tile)
Row = Floor((Tile origin Y – Point of interest Y) / Ground height of
a tile)
```

- Point of interest X/Y – kartes punkts, kuram nepieciešams iegūt rastra attēlu, kura teritorijā atrodas šis punkts
- Tile origin X – slāņa ģeogrāfiskais sākuma punkts. Šis punkts tiek noteikts keša ģenerēšanas brīdī, un ir pieejams keša konfigurācijas failā, kurš savukārt ir pieejams vai nu fiziski <server_name>\arcgiscache\<layer_name>\Layers\conf.xml, vai arī caur REST interfeisu <server_name>/ArcGIS/rest/services/<layer_name>/MapServer?f=json.
- Ground width of a tile – bildes garums (vai platums) * izšķirtspēja konkrētajā līmenī. Līmeņa izšķirtspēja ir mainīgs lielums, kas mainās atkarībā no mēroga (līmeņa), kādā ir attēlota karte. Lai iegūtu datus visiem līmeņiem (kas parasti arī būs vajadzīgs), nepieciešams aprēķināt rastra datu apgabalu katram līmenim.

Izgūstamās teritorijas ģeometriju būtu ieteicams konvertēt uz ekstentu (vairāku punktu ģeometriju konvertēt uz 4 punktiem), lai atvieglotu izgūšanas procesu. Parastā gadījumā būs nepieciešams rastra attēlu komplekts teritorijai, nevis punktam, tāpēc funkciju nepieciešams piemērot, respektīvi, izsaukt katram līmenim 4 reizes „Point of interest" parametra vietā liekot ekstenta punktus. Tāpēc nepieciešama funkcija, kas izmantojot augstāk rakstīto formulu, spēj atgriezt sarakstu ar līmeņiem un tiem nepieciešamo rindu (min/max numurs) un kolonnu (min/max numurs) numuriem, tā veidojot „rāmi" ar bildēm, kas attiecas uz izgūstamo teritoriju.

Tātad, pirmkārt jāizveido funkcija, kas katram kartes līmenim atrod nepieciešamos bilžu numurus, un atgriež tos, piemēram, šādā formātā (pēdiņu vietā liekot atbilstošo skaitli):

```
{"levelNr":"","rowFrom":"","rowTo":"","colFrom":"","colTo":""}
```

Konkrētā rastra slāņa līmeņu sarakstu, tāpat kā pārējos formulai nepieciešamos parametrus iespējams atrast slāņa konfigurācijas failā. Zemāk paskaidrots, kā iegūt funkcijai nepieciešamos parametrus.

- Point of interest X/Y – šos parametrus var iegūt no ekstenta, kas pieder lietotāja izvēlētajai teritorijai
- Tile origin X/Y – parametri no konfigurācijas – tileInfo.origin.x (un y)
- Ground width of a tile – bildes garums (vai platums) * izšķirtspēja konkrētajā līmenī. Bildes garums un platums atrodams konfigurācijas failā – tileInfo.rows (un cols), savukārt izšķirtspēja ir katram līmenim sava – tileInfo.lods[levelNr].resolution

Ievietojot mainīgos funkcijā, iespējams iegūt sarakstu ar minimālo/maksimālo rindu/kolonnu numuriem visiem līmeņiem, kas veido rāmi ar bildēm, kas nepieciešamas lietotāja izvēlētās teritorijas nosegšanai. Piemēram, autora veidotajā funkcijā saraksts (serializēts JSON formātā) izskatās šādi:

```
{"levels":[
{"levelNr":0,"rowFrom":6,"rowTo":6,"colFrom":10,"colTo":10},
{"levelNr":1,"rowFrom":13,"rowTo":13,"colFrom":21,"colTo":21},
{"levelNr":2,"rowFrom":26,"rowTo":26,"colFrom":42,"colTo":42},
{"levelNr":3,"rowFrom":53,"rowTo":53,"colFrom":84,"colTo":84},
{"levelNr":4,"rowFrom":133,"rowTo":133,"colFrom":211,"colTo":211},
{"levelNr":5,"rowFrom":266,"rowTo":266,"colFrom":423,"colTo":423},
{"levelNr":6,"rowFrom":532,"rowTo":532,"colFrom":846,"colTo":847},
{"levelNr":7,"rowFrom":665,"rowTo":666,"colFrom":1057,"colTo":1058},
{"levelNr":8,"rowFrom":887,"rowTo":888,"colFrom":1410,"colTo":1411},
{"levelNr":9,"rowFrom":1331,"rowTo":1332,"colFrom":2115,"colTo":2117}
,
{"levelNr":10,"rowFrom":1774,"rowTo":1776,"colFrom":2821,"colTo":2823
},
{"levelNr":11,"rowFrom":2662,"rowTo":2664,"colFrom":4231,"colTo":4235
},
{"levelNr":12,"rowFrom":5324,"rowTo":5329,"colFrom":8463,"colTo":8470
}
]
```

Tālāk nepieciešams šos rindu/kolonnu numurus pārvērst, lai tie atbilstu ArcGIS servera rastra datu glabāšanas struktūrai (skat 1.3.2 nodaļu) – respektīvi, rindu un kolonnu numuri tiek pārveidoti uz heksadecimālo formātu, kā arī direktorijas numurs

vienmēr satur 8 simbolus, tātad esošajiem heksadecimālajam skaitlim jāpievieno kreisajā pusē nulles, lai kopā būtu 8 simboli.

Rindu un kolonnu numuru konvertēšanai var izmantot šādas C# funkcijas:

- ToString(x, 16) – konvertē skaitli uz heksadecimālo skaitīšanas sistēmu
- PadLeft(8, '0') – pievieno kreisajā pusē „0” simbolus, lai kopā būtu 8 simboli

Tātad būtībā risinājums šeit ir iet cauri visiem līmeņu/rindu/kolonnu numuriem, konvertējot tos uz attiecīgo formātu, ko izmanto ArcGIS rastra datu direktoriju shēma, un izmantojot, piemēram FileInfo.CopyTo .NET funkciju, pa vienam šos failus pārkopēt uz citu lokāciju, saglabājot līmeņu/rindu/kolonnu direktoriju struktūru. Papildus nepieciešams pārkopēt arī slāņa konfigurācijas failu, jo tas būs nepieciešams slāņa pieslēgšanai kartei (process aprakstīts 5.2 nodaļā).

Kad visi nepieciešami faili pārkopēti uz citu lokāciju, nepieciešams tos nogādāt uz ierīci. Lai gan ir iespējams veidot kādu servisu vai funkciju, kas veic saziņu ar ierīci un automātiski nokopē šos failus uz ierīces uzreiz pēc izgūšanas, tomēr autora pieredze rāda, ka ir pilnīgi pieņemami likt lietotājam pašam šos failus pārkopēt uz ierīci, kas ir ierasta prakse šādās lietotnēs, kā arī var uzlabot datu pārnešanas ātrumu, jo maksimāli daudz funkcionalitātes tiek pārnests servera pusē, pilnībā izslēdzot mobilo ierīci no datu izgūšanas.

5.2. Rastra slāņu pievienošana kartei

Rastra slāņu pieslēgšanā parasti tiek izmantoti dažādi servisi, kas darbojas tiešsaistē, un iespējas glabāt šos slāņus lokāli, kā arī attēlot tos, nav īpaši aprakstītas un izplatītas. Apskatot ESRI ArcGIS JS ietvara iespējas, neizdevās atrast internetā apliecinājumu, ka kādam būtu strādājošs piemērs. Ir atrodami vairāki piemēri ar citiem kartogrāfijas ietvariem (OpenLayers, LeafLet) [13] [14], bet konkrēti ESRI ArcGIS JS tika minēts kā viens no tiem, kuriem šādas iespējas neesot [15] (vismaz neveicot izmaiņas ietvara pirmkodā atbilstošajām funkcijām).

5.2.1. Risinājums

Neskatoties uz to, ka internetā nav atrodams apstiprinājums, ka ESRI ArcGIS JS atbalstītu lokālos rastra slāņus, tika veikt izmēģinājums, kā rezultātā tika konstatēts, ka ietvars spēj veiksmīgi pievienot slāni kartei arī tad, ja tas netiek padots caur ArcGIS serveri, bet atrodas lokāli lietotnes mapē.

Risinājums tiek veidots balstoties uz pieņēmumu, ka dati tiek izgūti ar metodi, kas izmantota 5.1 nodaļā un dati tiek glabāti tieši tādā formātā. Slāņa pievienošanai nepieciešams izmantot TiledMapServiceLayer slāņa tipu. Izmantojot piemērā doto jauna slāņa tipa definēšanas tehniku [16], tiek veidots jauns TiledMapServiceLayer slāņa tips, kuram nepieciešams pārrakstīt getURL() funkciju, kura normālā gadījumā atgriež URL uz karšu servisa direktoriju. Bezsaistes režīmā nepieciešams, lai funkcija atgriež saiti uz lokālo attēla glabāšanas vietu, balstoties pēc 3 parametriem:

- Zoom level – izmantotais mērogs, palielinājums
- Row – attēla x pozīcija
- Column – attēla y pozīcija

Balstoties uz zināšanām par rastra datu glabāšanas shēmu (1.3.2) var secināt, ka piemērā dotā getURL funkcija ir lieliski piemērojama arī bezsaistes gadījumā, jo satur nepieciešamos pārveidojumus, lai iegūtu korektu ceļu:

- `toString(16)` izmaina skaitli uz heksadecimālo skaitīšanas sistēmu
- `dojo.string.pad()` funkcija pievieno nepieciešamās nulles, tādējādi iegūstot korektu ceļu uz nepieciešamo attēlu.

Attiecīgi nepieciešams izmainīt arī sākuma daļu, kas norāda bāzes mapi. Līdz ar to, iegūstam apmēram šādu funkciju:

```
getTileUrl: function(level, row, col) {
```

```
  var url = "rastra/layer1/" +
        "L" + dojo.string.pad(level, 2, '0') + "/" +
        "R" + dojo.string.pad(row.toString(16), 8, '0') + "/" +
        "C" + dojo.string.pad(col.toString(16), 8, '0') + "." +
        "jpg";
      return url;
}
```

Turpinot definēt jauno slāņa tipu, nepieciešams norādīt vairākus parametrus:

- this.spatialReference - esri.SpatialReference klases instance. Šajā gadījumā wkid 3059, kas apzīmē Latvijas koordināšu sistēmu, kas visbiežāk tiek izmantota šajā darbā
- this.initialExtent – kartes sākuma ekstents
- rows – attēla pikseļu skaits rindai
- cols – attēla pikseļu skaits kolonnai
- dpi - dots per inch
- format : attēla formāts
- compressionQuality : saspiešanas kvalitāte
- origin - rastra datu sākumpunkts (x, y)
- lods – mēroga līmeņu saraksts, katrs elements satur 3 mainīgos:
- level – līmeņa numurs
- resolution – izšķirtspēja, šajā gadījumā kartes vienības katrā pikselī
- scale : mērogs konkrētajam līmenim (piemēram, vērtība „5000”, kas apzīmētu 1:5000 mērogu)

Šos parametrus iespējams iegūt no katram slānim pievienotā JSON konfigurācijas faila (5.1 nodaļā pie izgūšanas minēts, ka nepieciešams pievienot konfigurācijas failu).

Visu šo parametru norāde un getUrl funkcijas izveidošana ir viss kas ir jāizdara rastra slāņa definēšanai. Kad tas izdarīt, kodā jāizveido slāņa instance un jāpievieno kartei ar map.addLayer() funkciju.

5.3. Vektordatu izgūšana konkrētai teritorijai

Līdzīgi kā rastra datiem (skat. 5.1 ievadu), arī vektordatu slāņi parasti tiek izgūti no datubāzes kādai konkrētai teritorijai. Tālāk, novietojot šos datus uz ierīces, būtu jāspēj tos attēlot kartē un veikt darbības ar tiem. ArcGIS JS API funkcionalitāte vektordatu attēlošana no lokāliem failiem nav gluži standarta funkcionalitāte, tomēr ir iespējams atrast veidu, kā to izdarīt.

5.3.1. Risinājums

ESRI sistēmās ir 3 veidi, kā piekļūt datiem no programmēšanas viedokļa:

- Izmantojot ArcObjects (skatīt 1.6 sadaļu)
- Izmantojot REST interfeisu (skatīt 1.7 sadaļu)
- Pieslēdzoties pa tiešo bāzes datubāzei (šajā veidā nav pieejami ģeogrāfiskie dati un to funkcijas, toties ātrāk un vienkāršāk varam piekļūt ne-ģeogrāfiskajiem datiem, piemēram, parastajiem atribūtiem)

Lai gan ArcObjects piedāvā veidot daudz elastīgāku datu izgūšanu ar praktiski neierobežotām iespējam, tomēr REST interfeisa Query funkcija nodrošina lielisku saderību ar ArcGIS JS API, jo piedāvā eksportēt izvēlētos datus tieši JS API nepieciešamajā formātā.

Izpētot ArcGIS JS API varam secināt, ka tas spēj izmantot 3 formātu lokālos slāņus:

- KML
- CSV
- JSON

Savukārt Query funkcija piedāvā eksportu sekojošos formātos:

- HTML
- KMZ
- JSON

Kā redzams, JSON formāts ir vienīgais, kurš ir pieejams gan Query funkcijai, gan apstrādājams ar ArcGIS JS API ietvaru. ĢIS nozarē JSON failu formāts ir kļuvis gandrīz vai par standartu, līdz ar to būtu ļoti ieteicams izmantot tieši šo formātu gadījumos, kad nepieciešams ĢIS datus ievietot failā izmantošanai lokāli.

Query funkcija

Query funkcijai iespējams piekļūt, izmantojot URL:

<servera-adrese>/ArcGIS/REST/services/<karšu-servisa-nosaukums>/MapServer/<kartes slāņa id>/Query

tādā veidā atverot Query funkciju kādam konkrētam kartes slānim, pēc norādītā kartes slāņa identifikatora. Funkcijai ir gan tīmekļa interfeiss, kurā iespējams vaicājumu veidot grafiski, gan arī iespējams funkciju izsaukt izmantojot pareizi izveidotu URL adresi, kas satur nepieciešamos parametrus. Datu izgūšanai priekš lietotnes noteikti jāizmanto izsaukšana caur URL, jo nepieciešams procesu automatizēt un padarīt pietiekami dinamisku, lai varētu to izsaukt ar dažādiem parametriem.

Query funkcija atgriež datus skaidri definētā formātā ar šādiem atribūtiem: ([] apzīmē sarakstu, {} – JavaScript objektu)

- "displayFieldName" - attēlojamais datu lauks, šī darba ietvaros nav svarīgs
- "fieldAliases" : {} - objekts, satur key-value sarakstu ar lauku atšifrējumiem
- "fields" : [] – masīvs ar objektiem, kas satur datubāzes lauku nosaukumus, atšifrējumus, tipus, garumus
- "geometryType" - slāņa ģeometrijas tips
- "spatialReference" – slāņa telpiskā atsauce
- "features" : [] – saraksts ar objektiem, kur katrs objekts satur 2 atribūtus:
- "attributes" : {} – key-value saraksts, apzīmē objekta laukus un to vērtības
- "geometry" : {} – objekta ģeometrija punktu saraksta veidā

Query funkcija pārsvarā tiek izmantota salīdzinoši vienkāršu vaicājumu veikšanai ĢIS tīmekļa lietotnēs, un šajā gadījumā tās sniegtā funkcionalitāte ir pietiekama, jo tā piedāvā datus atlasīt 3 veidos – pēc ģeometriskām īpašībām, pēc lauku vērtībām, vai arī kombinējot abas metodes.

Atlase pēc atribūtu vērtībām

ĢIS sistēmās ierasta prakse ir pievienot slāņiem statisku ģeogrāfisku informāciju, piemēram, katram objektam norādīt pilsētu, kurā tas atrodas. Tas tiek darīts, lai uzlabotu veiktspēju bieži pieprasītiem ģeogrāfiskiem vaicājumiem. Šajā gadījumā tas lieliski noder priekš datu izgūšanas kādai konkrētai teritorijai.

Lai atlasītu pēc atribūtu vērtībām, nepieciešams izmantot Query funkcijas „where" atribūtu, kurā SQL ļoti līdzīgā sintaksē tiek norādīts lauka nosaukums un filtrējamo atribūtu vērtības, piemēram URL formātā parametrs izskatītos šādi:

&where=CITY=RIGA (piebilde: visam parametram nepieciešams izmantot Url.Decode metodi, lai „=" simboli būtu viennozīmīgi saprotami)

ArcGIS REST API satur vairākus parametrus, bet zemāk ir saraksts ar 3 parametriem, kuri būtu obligāti nepieciešami REST API pielietošanai, izgūstot datus pēc atribūtu vērtībām.

- outFields – saraksts ar atgriežamajiem laukiem visiem tiem objektiem, kas tiks atlasīti. Varam izmantot „*" simbolu līdzīgi kā SQL, vai arī veidot sarakstu, atdalītu ar komatiem, ja gadījumā slānim ir ļoti daudz lauku, un uz ierīces nav nepieciešams pārnest pilnīgi visus atribūtus.
- F – norāda formātu, kādā atgriezt datus. Kā jau augstāk minēts, optimālākais formāts šādā lietotnē varētu būt JSON, tāpēc šo parametru norādīsim kā &f=JSON, lai pēc tam varētu atgrieztos datus saglabāt un novietot uz ierīces.
- Where – norādā vēlamo slāņa atribūtu un tā vērtību.
- returnGeometry – true, jo vēlamies saņemt arī objektu ģeometrijas, lai lietotne varētu tās attēlot uz mobilās ierīces

Atlase pēc objekta ģeometrijas šķelšanās ar citu objektu

Tā kā ne vienmēr visi slāņi saturēs nepieciešamos atribūtus, kas apzīmē to ģeogrāfisko atrašanās vietu, tad tādos gadījumos ir nepieciešams veikt atlasi pēc objekta ģeometrijas jeb telpisko atlasi.

Telpiskā atlase ir mazliet sarežģītāka. Lai veiktu telpisko vaicājumu, funkcijas Query parametrā „geometry" nepieciešams norādīt ģeometriju, pēc kuras vēlamies filtrēt objektus. Līdz ar to mums nepieciešams veids, kā iegūt šo te ģeometriju. Piemēram, ja līdzīgi kā iepriekšējā piemērā vēlamies atlasīt tos objektus no konkrēta slāņa, kas atrodas Rīgas teritorijā, sākumā mums vajag atlasīt Rīgas teritorijas ģeometriju, izmantojot citu REST API pieprasījumu.

Tātad, tipiskā gadījumā ir zināmas 2 lietas – ģeogrāfisks objekts, kura teritorijai vēlamies iegūt datus, kā arī slānis, no kura vēlamies šo informāciju iegūt. Būtībā jāveic 2 galvenās darbības:

- 1. Pieprasījums, lai atlasītu galvenās teritorijas ģeometriju
- 2. Pieprasījums, lai atlasītu objektus no šīs teritorijas.

Sākumā jāveic REST pieprasījums, lai iegūtu ģeometriju teritorijai, no kuras tālāk vēlamies atlasīt objektus. Lai to izdarītu, nepieciešams izmantot šādus parametrus:

- returnGeometry – true, jo ģeometrijā šajā pieprasījumā ir pati būtiskākā informācija, ko vēlamies saņemt
- where – norādam, atribūtu un vērtību, pēc kusra vēlamies atlasīt teritoriju
- f – formāts, json
- outFields – šajā gadījumā nav tik svarīgi, jo tāpat tiks izmantota tikai ģeometrija. Var norādīt OBJECTID, lai tiktu atgriezts tikai šis lauks, samazinot atgriežamo datu apjomu

No izveidotā REST pieprasījuma tiek saņemta atbilde JSON formātā, kuras „feature" atribūts satur vienu objektu ar vajadzīgo ģeometriju.

Kad ir iegūta vajadzīgā ģeometriju, nepieciešams to padot kā parametra „geometry" vērtību otrajam REST pieprasījumam. Vēl viens būtisks parametrs šajā gadījumā ir ģeometriskās attiecības parametrs, kas ģeometriju attiecības veidu, piemēram:

- esriSpatialRelIntersects – poligonu šķelšanās
- esriSpatialRelContains - viens poligons pilnībā satur otru
- esriSpatialRelCrosses - līniju krustošanās
- esriSpatialRelEnvelopeIntersects – objektu ģeometriju ekstenti krustojas

Datu izgūšanas gadījumā ir 2 galvenie varianti – meklēt precīzi ģeometriju šķelšanos, vai arī meklēt pēc ģeometriju „envelope" jeb ekstenta šķelšanās. Lai gan ekstenta šķelšanās atgriezīs arī objektus, kas atrodas ne gluži meklētajā teritorijā, bet diezgan tuvu tai, tomēr tas ir daudz ātrāks veids, kā veikt telpisko atlasi. It īpaši, ja objektu ģeometrijas satur ļoti daudz punktu.

Query funkcijas atgriežamo objektu limits

Vēl viena problēma, ko nepieciešams risināt, ir Query funkcijas atgriežamo objektu skaita ierobežojums. Tas ir atšķirīgs katram slānim, tomēr pārsvarā ir robežās no 100-500. Ierobežojums ir uzstādīts kā optimālais lielums [17], lai neradītu pārāk lielu slodzi uz serveri un to nav ieteicams atslēgt vai palielināt, jo tas ietekmē datu

atlasi visā sistēmā. Līdz ar to ir nepieciešams šo problēmu apiet. Problēmas risinājums atkarīgs no tā, kādu atlasi veicam – pēc atribūtiem, vai arī pēc ģeometrijas.

Ja veicam atlasi pēc atribūtiem, ierobežojumu var apiet, sākumā atlasot OBJECTID sarakstu no datubāzes, izmantojot parastu datubāzes vaicājumu un izmantojot attiecīgo where parametru (lai iegūtu datus no konkrēta apgabala) un sakārtojot atgrieztos OBJECTID augošā secībā. Tālāk šo sarakstu sadalām mazākās daļās, un sūtam vairākus pieprasījumus šīm konkrētajām daļām, pievienojot REST API pieprasījumam where parametru:

OBJECTID >= {1} and OBJECTID <= {2}, kur {1} un {2} ir konkrētā pieprasījuma pirmais un pēdējais OBJECTID. Tādā veidā veicam vairākus pieprasījumus, pārklājot visus mūs interesējošos OBJECTID.

Atlasot pēc ģeometrijas, ierobežojuma apiešana ir sarežģītāka. Šajā gadījumā vairs nav iespējas izmantot parasto datubāzes vaicājumu, jo tas nespēj filtrēt ierakstus pēc to ģeometrijas. Līdz ar to nepieciešams izmantot ArcGIS REST API funkcionalitāti. Pēc izpētes nākas secināt, ka ArcGIS Server 9.3 versijā Query funkcija nesatur veidu, kā šo ierobežojumu apiet. Tomēr, 10.1 versijā Query funkcija satur parametru returnIdsOnly [18], kura uzstādīšana uz true nodrošina to, ka serveris atgriež tikai OBJECTID lauku vērtības, toties vairs neņem vērā atgriežamo ierakstu skaita ierobežojumu. Šādā veidā varam apiet REST API Query funkcijas ierobežojumu, un iegūt sarakstu ar visiem OBJECTID, par kuriem mums nepieciešams atlasīt informāciju. Tālāk, līdzīgi kā iepriekš aprakstītajā risinājumā priekš atlases ar atribūtu lauku, sakārtojam OBJECTID sarakstu un veicam vairākus analoģiskus REST pieprasījumus, lai iegūtu mums vajadzīgo datu komplektu, ko pēc tam apvienojam vienā failā.

5.4. Vektordatu attēlošana un glabāšana

Normālā situācijā (tiešsaistes režīmā) vektordatu attēlošanai karšu lietotnes izmanto karšu servisus, kuriem veic reāla laika pieprasījumus un saņem aktuālus datus. Bezsaistes iespējas šajā jautājumā nav īpaši aprakstītas.

Jāatrod veids, kā ar ArcGIS JS API iespējams šos JSON formātā eksportētos datu slāņus pievienot kartei un attēlot. Lokālo datu slāņu attēlošana nav gluži standarta funkcionalitāte kartogrāfijas ietvariem, tie vairāk ir paredzēti datu iegūšanai no tiešsaistes servisiem.

5.4.1. Risinājums

ArcGIS JS API atbalsta JSON vektordatu slāņu pievienošanu kartei. Tas notiek, izmantojot esri.layers.FeatureLayer klases objektu, kuram tiek padots FeatureCollection objekts. Šis objekts savukārt sastāv no 2 daļām - layerDefinition un featureSet. [19]

FeatureSet ir slāņa objektu (katrs objekts satur divus sarakstus - ar atribūtiem un ģeometrijas koordinātām) saraksts JSON formātā. Šeit iespējams izmantot no REST Query Layer funkcijas eksportētos datus (aprakstīts 5.3 nodaļā), iepriekš tos padodot funkcijai esri.tasks.FeatureSet, kas izveido featureSet no JSON formātā esošajiem datiem, piemēram:

```
// nolasa JSON no faila, kurš satur izgūtos datus
var json = setVariableFromJsonFile(configLayer.path);
var fs = new esri.tasks.FeatureSet(json);
```

LayerDefinition prasa definēt ģeometrijas tipu (esriGeometryPoint, esriGeometryLine, esriGeometryPoligon), kā arī tam nepieciešams norādīt lauku definīciju. Pētījuma praktiskajā daļā tiks noskaidrots, ka šajā definīcijā vienīgais lauks, kuru nepieciešams norādīt, ir OBJECTID lauks. Tas tāpēc, ka lielākā daļa iebūvēto JS API atlases funkciju darbojas izmantojot šo lauku. Pārējos laukus nav nepieciešams norādīt, tos saraksts ir pieejams pie katra no objektiem json failā, kurš satur izgūtos datus. Būtībā, LayerDefinition parasti būs diezgan konstants:

```
"fields": [{
            "name": "ObjectID",
            "alias": "ObjectID",
            "type": "esriFieldTypeOID"
          },
```

FeatureLayer pievienošana no lokāla JSON objekta izskatās apmēram šādi [19]:

```
var fs = new esri.tasks.FeatureSet(jsonFS);

    var featureCollection = {
        layerDefinition: {
          "geometryType": "esriGeometryPoint",
          "fields": [{
            "name": "ObjectID",
            "alias": "ObjectID",
            "type": "esriFieldTypeOID"
          },          ]
        },
        featureSet: fs
    };
// izveido FeatureLayer instanci
var featureLayer = new esri.layers.FeatureLayer(featureCollection);
// pievieno slāni kartei
map.addLayer(featureLayer);
```

Šādi iespējams kartei pievienot vairākus vektordatu slāņus. Katrs slānis tiek pievienots atsevišķi un secībā. Tālāk ar šo slāņu objektiem iespējams veikt ierastās darbības – atlasīt, apskatīt atribūtus u.tml.

5.5. Lokācijas noteikšana / GPS

Līdz ar HTML5 ieviešanu ir krietni atvieglojusies lokācijas noteikšana tīmekļa lietotnēm. HTML5 lokācijas noteikšana izmanto vairākas metodes:

- IP adrese
- Wi-Fi pieslēgums
- GPS

HTML5 lokācijas funkcijas atgriež koordinātas LonLat vienībās izmantojot WGS 84 (WKID: 4326, EPSG:4326) telpisko atsauci. Tā kā lietotnes karte un visi pārējie dati tiek projicēti Latvijas koordinātu sistēmā LKS92 (WKID: 3059, EPSG:3059), ir nepieciešams HTML5 atgrieztās lokācijas koordinātas konvertēt uz LKS92 telpisko atsauci, lai tās būtu iespējams attēlot kartē. Tiešsaistes režīmā ArcGIS JS API būtu pieejamas funkcijas koordinātu konvertēšanai starp šīm abām (un ne tikai) atsaucēm, bet bezsaistes režīmā ir pieejamas tikai 2 veidu konvertācijas – no WGS84 uz Web Mercator sistēmu un otrādi. Visi citi varianti izmanto servera puses funkcijas, kas bezsaistē nav pieejamas.

5.5.1. Risinājums

HTML5 ir pieejamas 2 funkcijas lokācijas iegūšanai, izmantojot GPS. navigator.geolocation.getCurrentPosition() iegūst pozīciju un to atgriež, savukārt navigator.geolocation.watchPosition() turpina noteikt pozīciju (līdz tiek deaktivizēts) un izsauc pozīcijas maiņas notikumu katru reizi, kas pozīcija ir būtiski mainījusies. [20]

Neatkarīgi no izmantotās funkcijas, ir nepieciešama JavaScript bibliotēka, kas spēj konvertēt atgrieztās koordinātas uz citu telpisko atsauci. Veicot meklēšanu internetā, kā pirmais šāda veida ietvars tiek uzrādīts Proj4js [21]. Jaunākā versija iznākusi pasen un izskatās ka projekts sen nav ticis attīstīts, tomēr, pievienojot ietvaru lietotnei un pārbaudot tā darbību nākas secināt, ka ietvars spēs nodrošināt risinājumu telpisko atsauču konvertēšanai.

Bibliotēka sevī nesatur daudz telpisko atsauču definīciju, bet tos ir iespējams manuālu pievienot, turklāt šai bibliotēkai ir spēja pašai sameklēt nepieciešamos definīciju failus, ja ir zināms atsauces identifikators WKT sistēmā. Tie tiek meklēti tīmekļa resursā http://www.spatialreference.org, kas satur lielu sarakstu ar dažādām

telpisko atsauču definīciju failiem, kurus iespējams izmantot, lai pārslēgtos no vienas atsauces uz otru.

Zemāk ir funkcija, kurš tiek izsaukta brīdī, kad ir noticis koordināšu iegūšanas notikums. Tā demonstrē, kādā veidā nepieciešams apstrādāt datus, lai pēc tam tos varētu atgriezt kā esriPoint klases mainīgo un izmantot kartē.

```
function zoomToLocation(location) {
      var esriPtLatLon = new
esri.geometry.Point(location.coords.longitude,
location.coords.latitude);
      var esriPtLKS = convertLatLonToLKS92(esriPtLatLon);
      addGraphic(esriPtLKS);
}
```

Funkcija datu konvertēšanai.

```
var source = new Proj4js.Proj("EPSG:4326");    // Longitude/Latitude
var dest = new Proj4js.Proj("EPSG:3059");     // LKS92


function convertLatLonToLKS92(esriPoint) {
      var p = new Proj4js.Point(esriPoint.x, esriPoint.y);   //any
object will do as long as it has 'x' and 'y' properties
      Proj4js.transform(source, dest, p);     //do the
transformation.  x and y are modified in place
            return new esri.geometry.Point(p.x, p.y, new
esri.SpatialReference({ wkid: 3059 }));
}
```

5.6. Standarta ģeogrāfiskās funkcijas

Lai gan ArcGIS JS API ir diezgan liels ietvars un tas piedāvā plašu funkciju klāstu, tomēr liela daļa no funkcijām veic pieprasījumus uz serveri un bezsaistē nav lietojamas. Funkciju aprakstos ne vienmēr ir konkrēti aprakstīts, vai dotā funkcija ir pieejama bezsaistē, tāpēc, tika praktiski pārbaudītas šīs funkcijas, atlasot noderīgākās, kas ir pieejamas lietošanai bez interneta pieslēguma un servera starpniecības.

5.6.1. Identify

Funkcija kartes objektu identificēšanai, norādot kartē meklēšanas laukumu jeb ekstentu. Tīmekļa lietotnēs parasti izmanto REST interfeisa Identify funkciju, tomēr bezsaistē ir pieejama alternatīva, izmantojot zīmēšanas rīkjoslu un extent.contains(point) funkciju – šī funkcija atgriež true, ja dotais punkts atrodas ekstentā. Balstoties uz ESRI kodu galerijā esošo piemēru [22], kurš dod iespēju identificēt punktus, varam pārveidot funkciju, lai tā spētu noteikt arī poligona un lietotāja zīmētā ekstenta krustošanos.

Sākumā nepieciešams izveidot rīkjoslas mainīgo un piešķirt tam notikumu uz ekstenta zīmēšanas beigām.

```
var identifyToolbar = new esri.toolbars.Draw(map);
dojo.connect(identifyToolbar, "onDrawEnd", findPolygonsInExtent);
```

Funkcija, kas izmanto extent.contains(point)funkciju, lai noteiktu poligona un ģeometrijas pārklāšanos.

```
function checkIfExtentSplitsGeometry(extent, geometry) {
      var pts = geometry.rings[0];
      for (var i=0; i<pts.length; i++) {
            var esriPoint = new esri.geometry.Point(pts[i], new
esri.SpatialReference({ wkid: 3059 }));
            if (extent.contains(esriPoint)) {
                  return true;
            }
      }
      return false;
}
```

Funkcija, kas tiek izsaukta uz rīkjoslas zīmēšanas beigu noteikumu, kas katram vektordatu slāņa objektam (tikai tam slānim, ko norādījis lietotājs).

```
function findPolygonsInExtent(extent) {
```

```
	// lietotāja norādītais feature layer slānis
	var idLayer = featureLayerList[csiLayerIndex];

	var graphics = idLayer.layer.graphics;
	var results = [];
	var graphic;
	for (var i=0, il=graphics.length; i<il; i++) {
		graphic = graphics[i];
		if (checkIfExtentSplitsGeometry(extent,
graphic.geometry)) {
			results.push(graphic);
		}
	}
	return results;
}
```

5.6.2. Search

Funkcija tiek izmantota, meklējot vektordatu slāņos objektus pēc atribūtiem. Tīmekļa lietotnēs zem Search funkcijas parasti slēpjas izsaukums uz REST interfeisa Find vai Query funkcijām. Bezsaistes režīmā ArcGIS JS API nepiedāvā alternatīvu šai funkcionalitātei, tomēr šeit pilnīgi pietiek ar parasto JavaScript funkcionalitāti, piemēram array.filter funkciju, kas. Lai nodrošinātu dinamisku vaicājumu veidošanu ar vairākiem laukiem, nepieciešams funkciju pielāgot, izsaucot to ciklā un katru reizi saglabājot rezultātu priekš nākamā izsaukuma. Zemāk ir funkcijas piemērs, kas kā ievaddatus saņem slāni, kurā veikt meklēšanu, un masīvu ar parametriem formātā {field: „”, value: „”}.

```
function getSearchResults (featureLayer, filterArray) {
	// sākumā rezultāts ir visi objekti
	var results = featureLayer.graphics;

	for (var i=0; i<filterArray.length; i++) {
		results = results.filter(function(g) {
			return g.attributes[filterArray[i].field] ==
filterArray[i].filter;
		});

		// ja jau ir tukšs, ejam ārā
		if (results.length == 0) {
			break;
```

```
            }
      }
      return results;
}
```

5.6.3. Area, Length

Ģeometrisko aprēķinu funkcijas. Tīmekļa lietotnēs šīs funkcijas parasti tiek izsauktas, izmantojot REST interfeisa GeometryService servisu, kas nodrošina ģeometrisko funkciju izpildi. Bezsaistes režīmā kā alternatīva ir pieejama ArcGIS JS API klase esri.geometry, kas satur dažas standarta ģeometrijas apstrādes funkcijas, kuru izpildei nav nepieciešams serveris.

Esri.geometry klases funkcija geodesicAreas, kura atgriež objekta ģeometrijas laukumu, aprēķiniem spēj izmantot tikai ģeometrijas, kuru koordinātas ir WGS 84 (WKID: 4326, EPSG:4326) (skatīt 1.4 nodaļu) telpiskajā atsaucē. Tā kā šajā darbā veidotā lietotne izmanto LKS92 (WKID: 3059, EPSG:3059) (skatīt 1.4 nodaļu) telpisko atsauci, ir nepieciešams konvertēt objektu ģeometrijas uz WGS 84, pirms ģeometrija tiek padotas geodesicAreas funkcijai kā parametrs.

Piemēram, funkcija, kas ievadā saņem veco poligonu, un izveido jaunu poligonu, konvertējot vecās koordinātas no LKS92 uz WGS84.

```
function convertGeometryFromLKSToLatLon(geometry) {
      // poligons LonLat sistēmā
      var poly = new esri.geometry.Polygon(new
esri.SpatialReference({wkid:4326}));

      // pieņemam, ka poligonam nav multipart
      poly.addRing([]);

      for (var i=0; i<geometry.rings[0].length; i++) {
            var esriPoint = new
esri.geometry.Point(geometry.rings[0][i]);
            poly.insertPoint(0, i, convertLKS92ToLatLon(esriPoint));
      }

      return poly;
}
```

Palīgfunkcija punkta konvertēšanai no LKS92 uz WGS84. Izmanto Proj4JS koordināšu konvertēšanas bibliotēku [21]

```
var source = new Proj4js.Proj("EPSG:4326");    // Longitude/Latitude

var dest = new Proj4js.Proj("EPSG:3059");     // LKS92
function convertLatLonToLKS92(esriPoint) {
      var p = new Proj4js.Point(esriPoint.x, esriPoint.y);
      Proj4js.transform(source, dest, p;
      return new esri.geometry.Point(p.x, p.y, new
esri.SpatialReference({ wkid: 3059 }));
}
```

Kad izveidots jaunais poligons WGS84 koordināšu sistēmā, varam to padot kā ieejas parametru geodesicAreas funkcijai.

```
function getArea(geometry) {
      // parametrus nepieciešams padot WGS84 koordināšu sistēmā
      var geoCoords = convertGeometryFromLKSToLatLon(geometry);

      // funkcijai padod masīvu ar poligoniem, un tā atgriež masīvu
ar platībām
      // normālā gadījumā parasti tiks padots un atgriezts 1 poligons
      var areas = esri.geometry.geodesicAreas([geoCoords],
esri.Units.KILOMETERS);

      return areas[0];
}
```

5.6.4. Nepieejamās funkcijas

Parasti ģeogrāfisko datu analīzes funkciju izsaukšanai tiek izmantots vai nu REST interfeisa ģeometrijas serviss (*geometry service*) modulis, vai arī tās tiek veidotas servera pusē, izmantojot ArcObjects. Dažās no šīm funkcijām ir speciāli veidotas klienta pusē ar tīmekļa tehnoloģijām (piemēram, ArcGIS JS API Geometry klase satur vairākas no šīm, skatīt 5.6.3 sadaļu) dažas ir iespējams apiet, izmantojot citus ietvarus (piemēram, Proj4JS telpisko atsauču pārveidošanai, skatīt 5.5 sadaļu). Tomēr, lielākā daļa no šīm funkcijām nav pieejamas, izmantojot tikai tīmekļa tehnoloģijas. Autors uzskata, ka tas varētu būt tāpēc, ka:

- Funkcijas ir pietiekami sarežģītas, un tāpēc ietvaru veidotāji nav vēlējušies tās veidot, izmantojot tīmekļa tehnoloģijām
- Esošie tīmekļa tehnoloģiju ietvari nav paredzēti izmantošanai bezsaistes režīmā, ietvaru veidotāji pieņem, ka šie ietvari tiks izmantoti sasaistē ar serveri, tātad būs pieejamas arī servera puses funkcijas un nav nepieciešams funkcijas pārrakstīt uz tīmekļa tehnoloģijām

- Šīs funkcijas veic diezgan nopietnus aprēķinus un ar mobilo ierīču jaudu varētu nepietikt, lai šīs funkcijas veiktu

Zemāk ir saraksts ar dažām svarīgām funkcijām, kam autors pagaidām nav atradis aizvietojumu, izmantojot tīmekļa tehnoloģijas:

- Ģeometrijas vienkāršošana (*simplify*) – ieejā saņem objekta ģeometriju un vienkāršo to, samazinot koordinātu skaitu. Tas atvieglo un paātrina aprēķinus.
- Ģeometrijas bufera ģenerēšana (*buffer*) – ieejā saņem objektu un buferjoslas platumu, un atgriež jaunu ģeometriju, kas ir kā buferzona ieejas ģeometrijai noteiktajā platumā
- Ceļu slāņa analīze (*routing*) – dažādas funkcijas, kas analizē ceļu ģeometriju slāni, piemēram, ātrākā ceļa atrašana, tuvāko objektu noteikšana, objektu atrašana, kas atrodas x minūšu braukšanas attālumā utt.

5.6.5. Secinājumi

Rezumējot šo sadaļu varētu teikt, ka tīmekļa tehnoloģijas nodrošina samērā vienkāršu ģeogrāfisku funkciju veikšanu, sarežģītākos un jaudīgākos aprēķinus tomēr atstājot pilnīgi servera pusē. Tieši šīs funkcijas ir tā lieta, kur būtiskas priekšrocības ir ArcGIS Android SDK ietvaram. Tas satur vairāk ģeogrāfisko datu analizēs funkcijas, kas piemērotas tieši bezsaistes režīmam, piemēram, ģeometriju šķelšanās noteikšana, šķēluma laukuma noteikšana, objektu ģeometriju apvienošana, telpisko atsauču pārveidošana u.tml.

REZULTĀTI UN DISKUSIJA

Darba sākumā ir sniegs ieskats par to, kas ir ĢIS (konkrētāk aprakstīts tieši ArcGIS produkts) un kādi ir tās galvenie elementi, galvenās datu struktūras, kā arī šajā darbā biežāk izmantotie termini.

Ir izveidots lietotņu veidu (tīmekļa, *native*, hibrīdlietotne) salīdzinājums gan globāli, gan ĢIS kontekstā, kā arī aprakstītie izmantojamie ietvari katram no izstrādes veidiem.

Darba galvenais rezultāts ir radītais apraksts ĢIS bezsaistes lietotņu problēmu risinājumiem, kas ir arī praktiski pārbaudīti, izveidojot lietotni.

Pirmkārt, ir atrasts, aprakstīts un pārbaudīts veids, kā izgūt rastra datus konkrētai teritorijai un pēc novietošanas ierīces atmiņā, spēt tos attēlot.

Otrkārt, ir atrasts, aprakstīts un pārbaudīts veids, kā izgūt rastra gan pēc atribūtu vērtībām, gan ģeometrijas īpašībām. Ir aprakstīts, kā iespējams šos datus attēlot kartē un veikt ar tiem dažādas darbības.

Ir aprakstīts risinājums vairāku standarta ĢIS lietotņu darbību veikšanai, piemēram, GPS punktu attēlošana, kartes objektu identificēšana, kartes objektu meklēšana pēc atribūtiem, ģeometriskie aprēķini (ģeometrijas garums, platība).

Kopumā ir izdevies sasniegt gaidīto mērķi, vienīgi kā trūkums šeit jāmin datu labošanas un sinhronizācijas apraksts, kurš nav izveidots. Balstoties uz esošajām zināšanām neizdevās atrast pietiekami efektīvu veidu, kā šo problēmu varētu risināt ar ArcGIS JS API, kurš praktiski nesatur nekādu atbalstu lokālo vektordatu labošanai un nosūtīšanai uz serveri.

SECINĀJUMI

Ar tīmekļa tehnoloģijām un ArcGIS JS API ir iespējams radīt ĢIS lietotni datu apskatei bezsaistes režīmā. Lai gan ir jāsamierinās ar vairāku ģeogrāfisko datu analīzes funkciju trūkumu, tomēr ir iespējams izveidot pietiekami plašu funkcionalitāti, kas spēj sniegt lietotājiem reālu labumu. Piemēram, kaut vai tikai iespēja ģeogrāfiskos datus attēlot no ierīces atmiņas ir pietiekami svarīga funkcionalitāte.

No visām ArcGIS JS API ģeogrāfiskajām funkcijām, ko parasti izmanto tiešsaistes ĢIS lietotnēs, apmēram 10-20%, ir dublētas klienta pusē, izmantojot tīmekļa tehnoloģijas un ir lietojamas bezsaistes režīmā. Vēl apmēram 10-20% ir pieejamas, izmantojot dažādus citus ietvarus vai arī pašam veidojot šīs funkcijas. Būtībā ir pieejamas pašās svarīgākās funkcijas, savukārt sarežģītāku ģeogrāfisku uzdevumu veikšanai šis ietvars nav īsti piemērots.

Drīzumā (šī gada vasarā) plānots izlaist ArcGIS Android SDK 10.2 versiju, kas beidzot ietvers bezsaistes datu labošanas funkcionalitāti. Darba ietvaros ir izpētītas iespējas un risinājumi, ko spēj sniegt ArcGIS JS API un tīmekļa tehnoloģijas kopumā, tomēr, pirms darba turpināšanas un jaunu problēmu risināšanas, autoraprāt, būtu vērts sagaidīt ArcGIS Android SDK jauno versiju, izpētīt to un izdarīt secinājumus.

IZMANTOTĀ LITERATŪRA

1. ESRI. [Tiešsaiste] [Citēts: 2013. gada 01. 06.] http://en.wikipedia.org/wiki/Esri.

2. ArcGIS. [Tiešsaiste] [Citēts: 2013. gada 20. 04.] http://en.wikipedia.org/wiki/ArcGIS#ArcGIS_10.x.

3. ArcGIS Server. [Tiešsaiste] [Citēts: 2013. gada 20. 04.] http://www.esri.com/library/brochures/pdfs/arcgis-server.pdf.

4. Feature class basics. [Tiešsaiste] [Citēts: 2013. gada 25. 04.] http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=feature_class_basics.

5. Cached tile service. [Tiešsaiste] [Citēts: 2013. gada 27. 04.] http://webhelp.esri.com/arcgisserver/9.2/dotnet/manager/publishing/static_map_svcs.htm.

6. Geographic_coordinate_system. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://en.wikipedia.org/wiki/Geographic_coordinate_system.

7. Spatial reference. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://www.sharpgis.net/post/2007/05/05/Spatial-references2c-coordinate-systems2c-projections2c-datums2c-ellipsoids-e28093-confusing.aspx.

8. Spatial Reference Systems. [Tiešsaiste] [Citēts: 2013. gada 26. 05.] http://gis.stackexchange.com/questions/26094/what-is-spatial-reference-system-in-laymans-language.

9. Choosing a mobile development strategy. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://blogs.esri.com/esri/arcgis/2013/03/19/choosing-a-mobile-development-strategy-2/.

10. Cache manifest. [Tiešsaiste] [Citēts: 2013. gada 04. 05.] http://www.html5rocks.com/en/tutorials/appcache/beginner/.

11. Export Tile Cache. [Tiešsaiste] [Citēts: 2013. gada 27. 04.] http://resources.arcgis.com/en/help/main/10.1/index.html#//001700000187000000.

12. Deconstructing the map cache tiling scheme part 2. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://blogs.esri.com/esri/arcgis/2008/01/31/deconstructing-the-map-cache-tiling-scheme-part-ii-working-with-map-caches-programmatically/.

13. OpenLayers offline storage. [Tiešsaiste] [Citēts: 2013. gada 07. 05.] http://openlayers.org/dev/examples/offline-storage.html.

14. LeafLet offline storage. [Tiešsaiste] [Citēts: 2013. gada 07. 05.] https://gist.github.com/leplatrem/1415767.

15. Cache tiles in WebSQL instead of LocalStorage. [Tiešsaiste] [Citēts: 2013. gada 07. 05.] http://forums.arcgis.com/threads/48721-Cache-tiles-in-WebSQL-instead-of-LocalStorage.

16. Custom tiled layers. [Tiešsaiste] [Citēts: 2013. gada 25. 04.] http://help.arcgis.com/en/webapi/javascript/arcgis/jssamples/layers_custom_tiled.html .

17. ArcGIS REST API Query limits. [Tiešsaiste] [Citēts: 2013. gada 23. 05.] http://blog.davebouwman.com/2009/02/21/displaying-large-selection-sets-with-the-esri-rest-api/.

18. ArcGIS REST API 10.0. [Tiešsaiste] [Citēts: 2013. gada 23. 05.] http://help.arcgis.com/en/arcgisserver/10.0/apis/rest/.

19. FeatureLayer from JSON. [Tiešsaiste] [Citēts: 2013. gada 29. 04.] http://forums.arcgis.com/threads/48925-FeatureLayer-from-JSON.

20. Using geolocation. [Tiešsaiste] [Citēts: 2013. gada 26. 05.] https://developer.mozilla.org/en-US/docs/WebAPI/Using_geolocation.

21. Proj4JS. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://trac.osgeo.org/proj4js/.

22. Graphics Extent Query. [Tiešsaiste] [Citēts: 2013. gada 26. 05.] http://help.arcgis.com/en/webapi/javascript/arcgis/jssamples/graphics_extent_query.html.

23. REST Query Layer. [Tiešsaiste] [Citēts: 2013. gada 28. 04.] http://resources.esri.com/help/9.3/arcgisserver/apis/rest/.

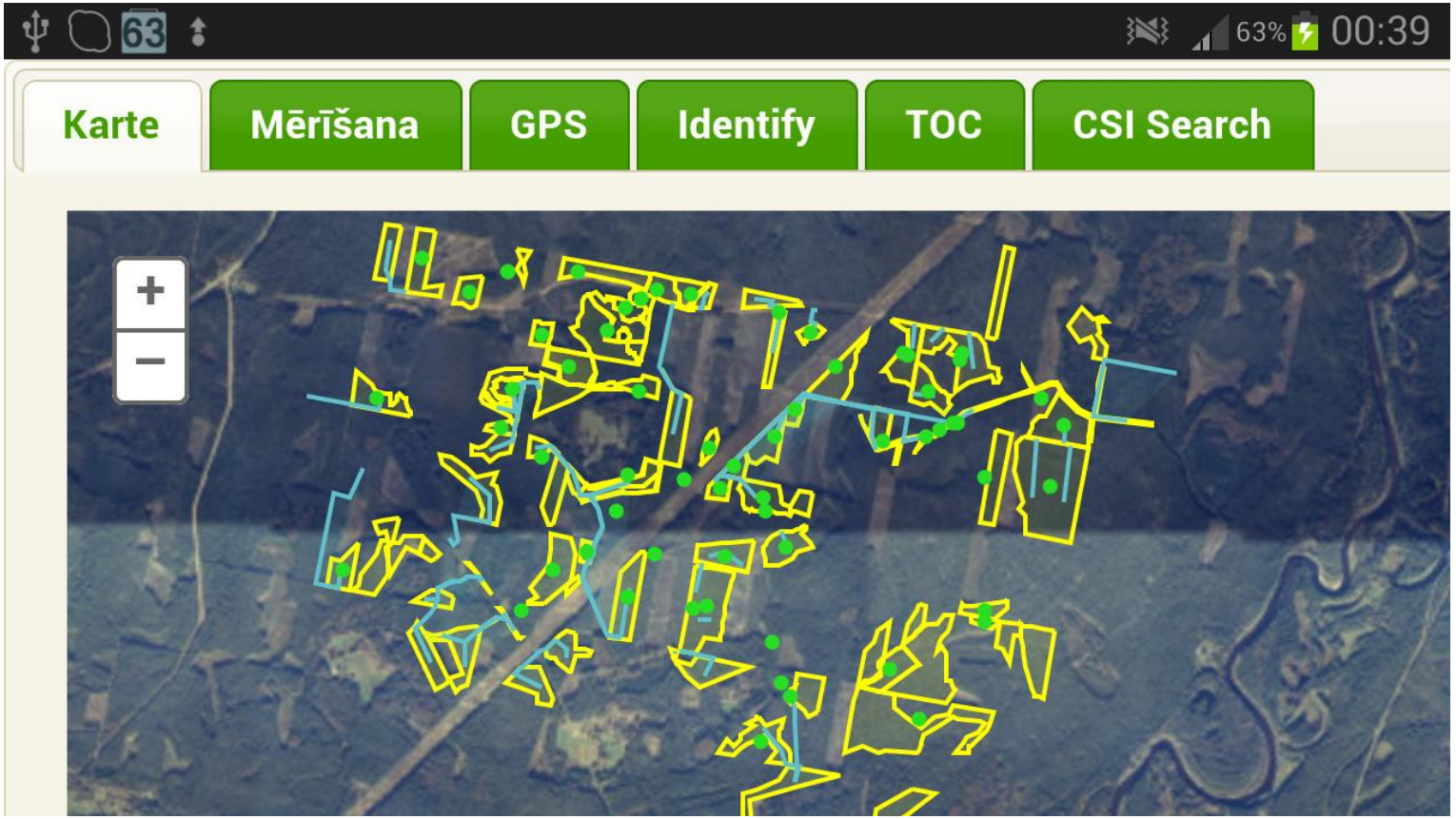
24. [Tiešsaiste] [Citēts: 2013. gada 28. 04.] http://gis.stackexchange.com/questions/8791/featurelayer-creating-in-javascript-arcgis-api.

25. ArcGIS REST API Query limits. [Tiešsaiste] [Citēts: 2013. gada 23. 05.] http://blog.davebouwman.com/2009/02/21/displaying-large-selection-sets-with-the-esri-rest-api/.

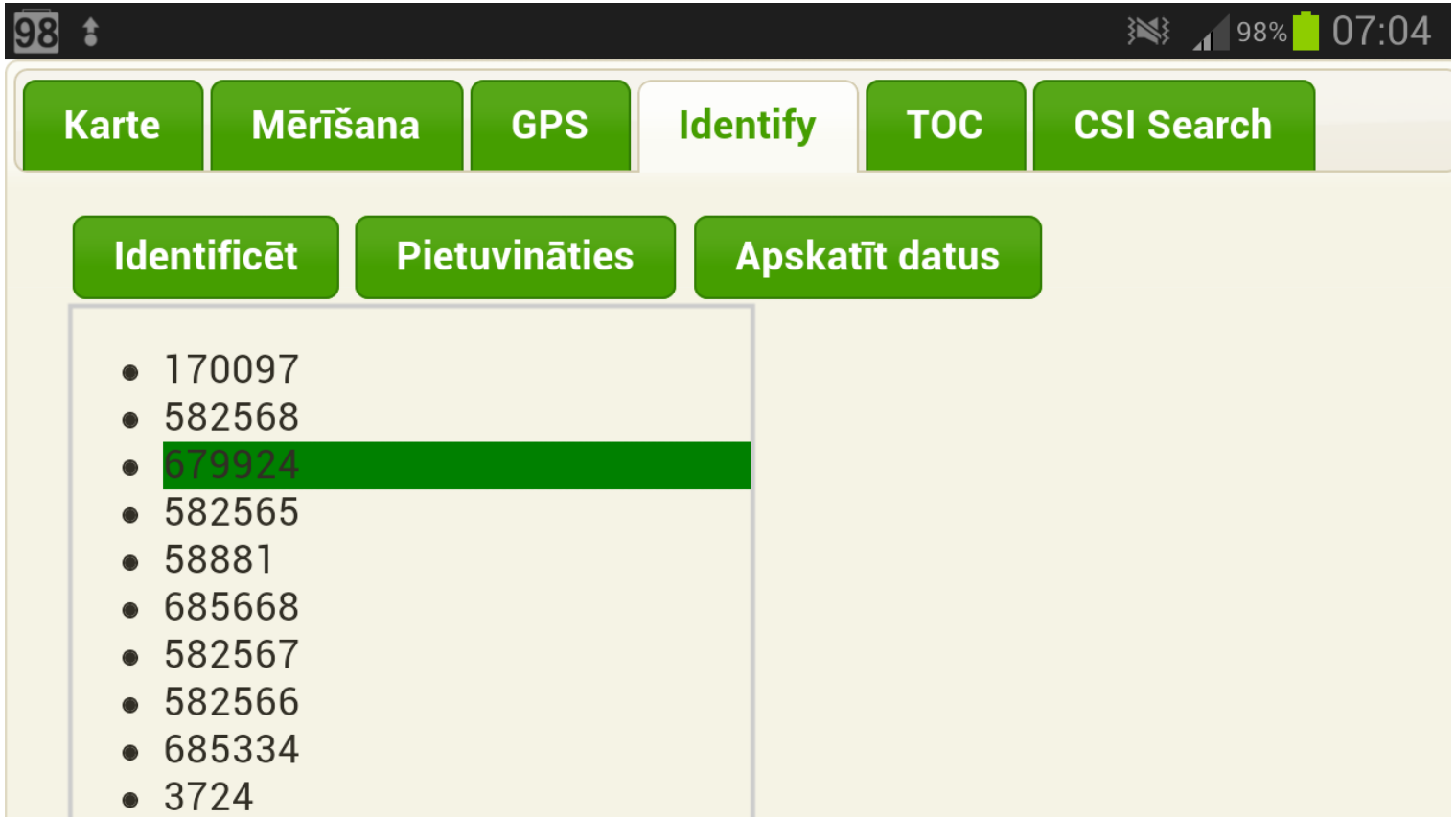
26. Choosing a mobile development strategy. [Tiešsaiste] [Citēts: 2013. gada 01. 05.] http://blogs.esri.com/esri/arcgis/2013/03/19/choosing-a-mobile-development-strategy-2/.

PIELIKUMI

1. PIELIKUMS - LIETOTNES KARTES SKATS



2. PIELIKUMS - OBJEKTU IDENTIFICĒŠANAS CILNE



DOKUMENTĀRĀ LAPA

Bakalaura darbs „Android lietotnes izveide ĢIS datu apstrādei bezsaistes režīmā” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Andris Skudra

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: asociētais profesors Dr.sc.comp. Ģirts Karnītis

Recenzents: docents Dr.sc.comp. Maksims Kravcevs

Darbs iesniegts Datorikas fakultātē _____.

(iesniegšanas datums)

Dekāna pilnvarotā persona:

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____

(darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(sekretāra paraksts)