TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andrei Gužovski IADB185818

Parkimisalade otsimise platvormi arendus Tartu näitel

Bakalaureusetöö

Juhendaja: Aleksei Talisainen
Magistrikraad
Kaasjuhendaja: Ilja Gužovski
Magistrikraad

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Andrei Gužovski

17.05.2021

Annotatsioon

Selle lõputöö eesmärk on lahendada parkimisruumi otsimise probleem, arendades parkimisruumi uurimise, otsimise ja navigeerimise Androidi rakendust. Lahendus on arendatud agiilse metoodikat kasutades ning põhineb Tartu ArcGISi geograafilise infosüsteemi platvormil, mille on välja töötanud linnavalitsus.

Lahendus on prototüüp-rakendus valmis testimiseks ja edasiseks arenguks ning avatud laiendamiseks teistes ArcGIS teenustega seotud valdkondades. Rakendus võimaldab uurida, otsida parkimisalasid ja eriparkimiskohtasid ning nendeni navigeerida. Lisaks on rakendus suuteline looma, redigeerima ja eemaldama ArcGIS platvormi andmeid eesmärgiga koguda ja parandada ArcGIS teenuse kvaliteeti.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 18 joonist, 3 tabelit.

Abstract

Parking Space Search Platform Development on the Example of Tartu

The purpose of this thesis is to solve the problem of searching for parking space by developing an Android application for exploring, searching, and navigating to parking space. The solution is designed using agile methodologies and is based on the example of Tartu and ArcGIS geographic information system platform developed by the city government.

The solution is prototype-ready for testing and further development as well as open for expansion to use in other fields related to ArcGIS services. The Android application allows you to explore, search and navigate to parking lots and special parking places. In addition, the application is capable of creating, editing, and removing ArcGIS platform data with the goal of gathering and improving ArcGIS service quality.

The thesis is in Estonian and contains 27 pages of text, 5 chapters, 18 figures, 3 tables.

Lühendite ja mõistete sõnastik

MVP	*Minimum viable product*, minimaalne elujõuline toode
OS	*Operating system*, operatsioonisüsteem
API	*Application programming interface*, rakenduse programmeerimise liides
MVC	*Model-View-Controller* arhitektuur
MVP	*Model-View-Presenter* arhitektuur
MVVM	*Model-View-ViewModel* arhitektuur
UI	*User interface*, kasutajaliides
Dependency	Teek millest rakendus on sõltuv
Epic	Agiilses metoodikas töö ühik
User Story	Agiilses metoodikas e*pic*u alamühik

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lõputöö peamine eesmärk luua töötav Androidi rakendus agiilse metoodika abil. Tulemuseks saab loodud MVP, mis võimaldaks spetsiaalselt Tartu parkimisalade andmete kogumist ja kasutamist, mida saaks ka edasiseks arendamiseks kasutada.

Selle eesmärgi saavutamiseks viiakse läbi võimalike nõuete, tehnoloogiate ja arhitektuuride analüüs ning olemasolevate lahenduste ülevaade. Kogutud teadmised peaksid pakkuma optimaalset lahendust eesmärgi täitmiseks.

1.1 Taust ja motivatsioon

Tänapäeval on laialt kasutusel GIS tehnoloogiad ja ruumilised andmed ning nende andmete kogumine läbi nutiseadmete on tavakasutajale märkamatu. Teenused nagu Google Maps, Waze, Apple Maps kasutatakse igapäevaselt navigeerimise ja kaartide vaatamise jaoks. Need rakendused pakkuvad ulatuslikult andmeid aadresside, teede ja huvipunktide kohta, kuid parkimisalade andmed võrreldes nendega on vähem detailsed ning paljud parkimisalad puuduvad kaartidel üldse.

1.2 Probleemi püstitus ja aktuaalsus

Parkimisalade andmete puudus või ebatäpsus tekitab mitmeid probleeme:

- Parkimiskoha leidmine võib muutuda keeruliseks ja aeglaseks nii kohaliku kui ka mitte kohaliku elaniku jaoks.
- Kulleriteenuste kvaliteet langeb, kuna navigatsioonirakendust kasutavad kullerid kulutavad tihti aega, et leida kliendi asukohale kõige lähim parkimiskoht, sest navigatsioonirakendused ei pruugi suunata parklani.
- Hea parkimisalade andmestiku puudumine, võib tekitada linnavalitsusel probleeme infrastrukuuri arendamist puudutavate otsuste tegemisel.

Selle lõputöö raames proovitakse neid probleeme lahendada Tartu linna näitel, kuna Tartu linnvalitsus on viimastel aastatel pidevalt oma infrastruktuuri arendanud.

1.3 Lõputöö eesmärgid

Enne arenduse alustamist viiakse läbi täielik probleemi uurimine, mille käigus leitakse lahendused parkimisalade andmete saamisele ja nõuete kogumisele.

Sellele järgneb turul olemasolevate lahenduste ja rakenduse realiseerimiseks võimalike tehnoloogiate ja arhitektuuride uurimine.

Arendamisetapil kasutatakse agiilsele metoodikale omaseid elemente. Siiski luuakse ka arendusprotsessi dokumentatsioon.

Tulemuseks saab loodud MVP lahendus parkimisalade andmete kogumiseks ja kasutamiseks, mis lahendab püsistatud probleemid.

Arenduse lõpus toimub võimalusel koostöö arutamine Tartu linnavalitsusega.

2 Probleemi analüüs

Populaarsemad navigeerimise rakendused ei paku täielikke ja detailseid andmeid parkimisalade kohta, mis muudab nende otsimise problemaatiliseks, näiteks kui on vaja leida õige parkimise koht maja juures. Rakendus, mis võimaldaks nende kiire otsimise ning annaks lõppkasutajatele võimaluse lisada ning uuendada andmed, lahendaks selle probleemi. Lisaks sellele, võiks selline rakendus leida kasutust veel kahel juhul:

- Infrastruktuuri arendamine nõuab ulatuslikke ruumiliste andmete analüüsi, sealhulgas parkimisalade ja -kohtade arv ühe linna elaniku peale. Rakendus, mis võimaldaks tavakasutajatel parkimisalade andmete lisamist, võiks olla kasutusel linnavalitsuse või Eesti avaandmete teenuse informatsiooni kiiremaks ja täpsemaks kogumiseks.

- COVID-19 pandeemia mõju tõttu toidu tellimise ja kohaletoimetamise teenuste kasutamine on saanud populaarsemaks ja see jätkub tänaseni positiivses trendis [1]. Selleks, et võita kliente, peab olema tagatud teenuse kõrge kvaliteet, näiteks kasutades parkimisalade andmeid saaks kuller leida sobiva parkimiskoha, millega jõuaks toode kõige kiiremini tarnijani.

Probleemi optimaalse lahenduse leidmiseks on vaja:

- leida parim viis geoandmete kogumiseks ja salvestamiseks ning uurida olemasolevat informatsiooni.

- määrata kindlad nõuded, mille järgi valida kõige sobivamad tehnoloogiad ja arhitektuur rakenduse loomiseks.

2.1 Olemasolevad andmed

Olemasolevate andmete oleku uurimiseks on koostatud kiri Tartu linnavalitsusele, küsides abi lõputöö jaoks vajalikke andmete ehk parkimisalade andmete leidmisega. On

saadud vastus, et Tartu koostas äsja tasulise parkimisala skeemi ArcGIS platvormil, mis asub Tartu Linnavalitsuse GIS portaalis [2].

Autori arust tõestab sellise rakenduse olemasolu veelgi rohkem parkimisalade geoandmete tähtsust. Antud rakendus on ArcGIS platvormi põhjal JavaScript veebirakendus, mis võimaldab filtreerida kaardil olevaid andmeid ning kiiresti neid otsida aadressi või hetkese asukohta järgi. Selle funktsionaalsus andis algse arusaamise, mis võiksid olla nõuded rakenduse loomisel.

2.2 Nõuded

Rakenduse loomise nõueteks on:

- Rakendus peaks olema arendatud võimalikult kiiresti ja võimalikult väiksete inimressursside kasutamisel.
- Rakenduse arenduse platvormiks on Android, kuna Androidi nutiseadmed on levinumad kui iOS ning JavaScripti veebirakendus on Tartu poolt juba loodud.
- Rakendus peaks olema sobitatav teiste rakendustega või laiendatav.
- Rakendus peaks kasutama võimalikult kaasaaegseid tehnoloogiaid.

2.3 Olemasolevate lahenduste analüüs

Enne rakenduse osaga alustamist peab autor võrdlema, kuidas olemasolevad lahendused toimivad parkimisalade andmete otsimisel. Kuna selle lõputöö raames rakenduse arendamine toimub Androidis, siis võrdleb autor nelja kõige sarnast lahendust, mis on leitud Google Play Store'ist:

- Google Maps
- Waze
- Sygic
- QField

2.3.1 Google Maps

Tartu piirkonnas Google Maps „parking lot“ otsingu järgi näitab kaart ainult 10 parkmisala punkti. Need punktid ei näita täpselt parkimisalade pindalad ega kohtade arvu.

Google My Maps veebirakendusel on võimalik luua oma kaardid, lisada nendele nii punktid kui ka kujud ning need lahti teha Google Maps Android rakenduses. Oma kaardil olevate punktide ja kujude järgi saab navigeerida ka, kuid Android rakendusest uusi geonandmed kaardile lisada ei ole võimalik. Google Maps rakendusega ei saa kasutada ArcGIS platvormi kaarte. [2]

2.3.2 Waze

Waze turundab ennast kui GPS navigeerimist rakendust, mida on võimalikult mugav kasutada autos. Waze „Tartu parking lots“ otsingu järgi näitab kaart ainult üheksat parkmisala punkti, kus asuvad mingisugused parklad. Nende punktide järgi ei ole täpselt parkimisalade pindala näha. Nende punktide lisainfo järgi ei saa teada ka kohtade arvu. Waze ei paku oma loodud kaartide kasutamist ega ArcGIS platvormi kaartide kasutamise võimalust. [3]

2.3.3 Sigyc

Sygic turundab ennast kui ilma võrguühenduseta navigeerimise rakendust. Sygic pakub võimalust eellaadida endale meelepärase piirkonna või riigi kaardi, kuid mitte oma tehtud kaardi ja mitte ArcGIS platvormi kaardi. Kaartidele punktide ja kujude lisamise võimalus puudub. Tartu piirkonnas Sygic „parking“ otsingul leiab 29 parkmisala. Lisainfost saab leida parkimisala tunnitasu, kui parkla on tasuline, ning tasuta parkimise aja pikkust. [4]

2.3.4 QField

QField on ehitatud professionaalse QGIS-i avatud lähtekoodiga projekti peale, mis võimaldab kasutajatel oma QGIS platvormi kaarte kasutada. QField ei paku parkimisalade otsingu võimalust, kui kasutuseks valitud kaart ei sisalda parkimisalade geoandmeid. Samas QField ei anna võimlalust kaarti netist laadida, selle kaardi, mida kasutaja tahab kasutada, peab eelnevalt laadima oma nutiseadmesse. Kui kaart on eellaaditud ja valitud, siis võimaldab QField nii kaardile uute geoandmete lisamist kui ka olemasolevate redigeerimist. Navigeermise funktsionaalsus QFieldil puudub. [5]

3 Tehnoloogiate ja arhitektuuride analüüs

3.1 Programmeerimis keelte valik

Tänapäeval on palju erinevaid programmeerimiskeeli, mida võib Androidi rakenduse loomiseks kasutada. Üldjuhul mobiilrakenduste arendamisel võib neid jagada kaheks tüüpiks: mitme platvormi lahendused, mille abil saab luua rakendust mitme OS-i jaoks, ja ühe platvormi lahendused ehk millega saab luua ainult ühe OS-i rakenduse.

Mitme platvormi lahenduste valikus on React Native raamistik ja Flutter kasutajaliidese tarkvaraarenduskomplekt, mis mõlemad annavad võimaluse luua rakenduse, mis töötaks iOS-il, Androidil ja veebirakendusena.

Ühe platvormi lahenduseks on Java või Kotlin programmeerimiskeelega kirjutatud Androidi rakendus.

Androidi natiivne rakendus pakub paremat jõudlust nutiseadme kasutamisel ning paremat kasutajakogemust. Samas Androidi nutiseadmed on kõige levinumad nutiseadmete turul. [6]

Nõude "Rakendus peaks olema arendatud võimalikult kiiresti ja võimalikult väiksete inimressursside kasutamisel." Täitmiseks sobib kõige paremini ühe platvormi lahendus, mis jätab valiku – kas Java või Kotlin?

Selle lõputöö raames on autor valinud Kotlini, vastavalt nõuele „Rakendus peaks kasutama võimalikult kaasaaegsemaid tehnoloogiaid.“, sest Kotlin on Google’i poolt soovitatav keel Androidi rakenduste loomiseks. [7]

3.2 Geoinfosüsteemide valik

3.2.1 QGIS

QGIS on avatud lähtekoodiga geoinfosüsteem, mis on tuntud oma lihtsuse ja universaalsuse poolest ning mis sobib hästi väikeste projektide ja ettevõtete jaoks. Samas

QGIS toetab mitut andmete formaati, sealhulgas ArcGIS formaati. Antud lõputöö raames võiks kaaluda QGISi kasutamist aga kuna QGIS ei paku Androidi jaoks rakenduse programmeerimise liidest, mis tunduvalt raskendab ja aeglustab Androidi rakenduse loomist, siis lõpp rakenduse jaoks on valitud olemasolev Tartu ArcGIS platvorm. [8]

3.2.2 ArcGIS

ArcGIS on geoinfosüsteem, mida haldab Environmental Systems Research Institute (Esri). ArcGIS pakub eelloodud arhitektuuri kaartide loomiseks ja geograafiliste andmete jagamiseks organisatsiooni sees või avalikult veebis. [9]

ArcGIS ärilise kasutamise jaoks peab ostma litsentsi. ArcGIS pakub omalt poolt väga ulatuslikke võimalusi litsentsi seadistamiseks. Samas ArcGISi võib vabalt kasutada tarkvara arendamise jaoks ning see pakub rakenduse programmeerimise liidest mitmete süsteemide ja programmeerimiskeelte jaoks, sealhulgas Androidi ja Kotlini jaoks.

„Rakendus peaks olema sobitatav teiste rakendustega või laiendatav.“ nõude täitmiseks sobib kõige rohkem just ArcGIS kuna Tartu linnavalitsusel on oma ArcGIS platvorm, mille abil on lihtne seadistada Androidi rakendust, et see kasutaks olemasolevas süsteemis olevad andmed.

3.3 Arhitektuuride valik

Androidi rakenduse arhitektuur erineb tavalise veebirakenduse arhitektuurist selle poolest, et Androidi rakendusel on mitu sisenemispunkti, samal ajal kui veebirakenduse sisenemispunktiks on selle põhine aadress. Näide:

Androidi meili rakenduse sisenemispunktideks on postkast ja uue kirja loomine. Esimesel juhul kasutaja lihtsalt avab rakenduse, siis talle on näha postkast. Teisel juhul, kui kasutaja vajutab kuskil teises rakenduses või veebilehel mingi meiliaadressi peale, siis Android pakub millise rakendusega see avada ning kui on valitud meili rakendus, siis rakendus avab uue kirja loomise akna. Meili veebirakenduse puhul ainukeseks sisenemispunktiks on postkast, pärast mida on võimalik valida järgmine tegevus.

Androidi erinevaid sisenemispunkte tavaliselt nimetatakse Activity'ks. Tavaliselt Activity sisaldab Fragmente, mis on Activity loogilised alamosad. Activity ja Fragmenti kasutamise juhtum sõltub täpsemalt valitud arhitektuurist.

Androidi rakenduste kolmeks kõige levinumaks arhitektuuriks on: MVC, MVP ja MVVM. Rakenduse arhitektuuri kasutamise põhiülesanneteks on erinevate rakenduste osade eristamine ja nende taaskasutamise lihtsustamine. Selles peatükkis toimub nende ülevaade ning rakenduse jaoks parima võimaliku arhitektuuri analüüs.

3.3.1 MVC – Model-View-Controller

Androidi API-d olid algselt loodud arhitektuuri nimega Model-View-Controller või MVC ja see on siiamaani põhiline arhitektuur. MVC-s peavad kõik rakenduse objektid olema mudel-, vaate- või kontrollerobjektid.

- Mudelobjektid sisaldavad rakenduse andmeid ja äriloogikat. Mudelklassid on tavaliselt kavandatud modelleerima rakendusega seotud asju, näiteks GIS rakenduse puhul geoandmed. Mudelobjektidel pole kasutajaliidese kohta teadmisi; nende ainus eesmärk on andmete hoidmine ja haldamine.

- Vaateobjektid teavad, kuidas end ekraanile joonistada ja kuidas reageerida kasutaja sisendile, näiteks puudutustele. Lihtne reegel on see, et kui seda on ekraanil näha, siis see on vaade.

- Kontrollerobjektid seovad vaate ja modelleerivad objektid omavahel. Need sisaldavad rakenduse loogikat. Kontrollerid on loodud reageerima erinevatele vaateobjektide käivitatud sündmustele ning haldama andmevoogu mudeliobjektide ja vaatekihi vahel. Androidi puhul tavaliselt kontrollerobjektid on nimetatud Activity või Fragment.

MVC sobib suurepäraselt väikeste lihtsate rakenduste jaoks, mida on vaja kiiresti arendada. Üldiselt on vaja hoida rakenduse kood puhas vastavalt Clean Code ja SOLID printsipiidele. Suuremates ja keerukamates rakendustes võib kontrollerikiht muutuda väga suureks ja keerukaks, kui see toimub, siis üldjuhul peab valima mingi teise arhitektuuri oma rakenduse jaoks. [10, pp. 37-45]

3.3.2 MVP – Model-View-Presenter

MVP arhitektuur on üks võimalustest suurema rakenduste osad veelgi rohkem eristada ning selleks on kasutuses Presenter.

Presenter vastutab vaate ja mudeli vahelise tegutsemise eest. See otsib mudeli andmed ja tagastab need vormindatud vaatena. Kuid erinevalt tüüpilisest MVC-st otsustab ka see, mis juhtub, kui vaatega suheldakse. [11]

3.3.3 MVVM – Model-View-ViewModel

Android MVVM arhitektuuri puhul mudelobjekt jääb saamaks ja vaate loogika ehk UI loogika liikub Activity ja Fragment objektide sisse.

ViewModeli objekt pakub konkreetse kasutajaliidese komponendi andmeid, näiteks Activity või Fragment, ja sisaldab mudeliga suhtlemiseks andmetöötluse äriloogikat. Näiteks võib ViewModel andmete laadimiseks teavitada teisi komponente ja edastada neile kasutajate taotlusi andmete muutmiseks. [10, pp. 367-368]

MVVM arhitektuur on Google'ga soovitatav arhitektuur ning Androidis on olemas kõik vajalikud klassid selle realiseerimiseks. [12]

3.3.4 Arhitektuuride võrdlus

ArcGIS Runtime API Android ei anna mingit erilist juhendid sellele, mis arhitektuur sobiks rohkem. Mõnedes Esri näidetes kasutatakse MVP, aga autori arvates MVVM arhitektuur sobib kas sama hästi või isegi rohkem [13]. MVVM arhitektuur:

- võimaldab Clean Code ja SOLID printsiipide jälgimist; [15]
- võimaldab ViewModelis võrgu kasutamisel andmete asünkroonset laadimist ilma kasutajaliidese seoseta ja samas nende andmete järjestikust laadimist enne kui kasutajaliidese elemendid võtavad need kasutusele;
- on enim levinud arhitektuur Androidi platvormil, mis lihtsustab rakenduse laiendamise ja hooldamise.

4 Rakendamine

Agiilse metoodika abil rakenduse loomisel kasutab autor selle tööraames selliseid agiilseid mõisteid nagu äriprotsess, *epic*ud ja kasutajalood. Tabelitel 1 ja 2 on välja toodud põhilised rakendusele vajalikud funktsioonid, ning just nende järgi toimub arendamine. Arendamise protsessi lihtsustamiseks on tehtud äriprotsessi diagramm.

Tabel 1 Epicud

Epic ID	**<sellise tüübi> kasutajana**	**Ma tahan <täita mõnda ülesannet>**	**et saaksin <mingi eesmärgi saavutada>**
1	Rakenduse kasutajana	Ma tahan uurida kaarti	et saaksin näha kõiki parkimisalasid selles piirkonnas.
2	Rakenduse kasutajana	Ma tahan navigeerida kaardil	et saaksin leida ja jõuda täpse parkimiskohani.
3	Rakenduse kasutajana	Ma tahan lisada kaardile andmeid	et saaksin aidata koguda andmeid riigi avaandmetesse.

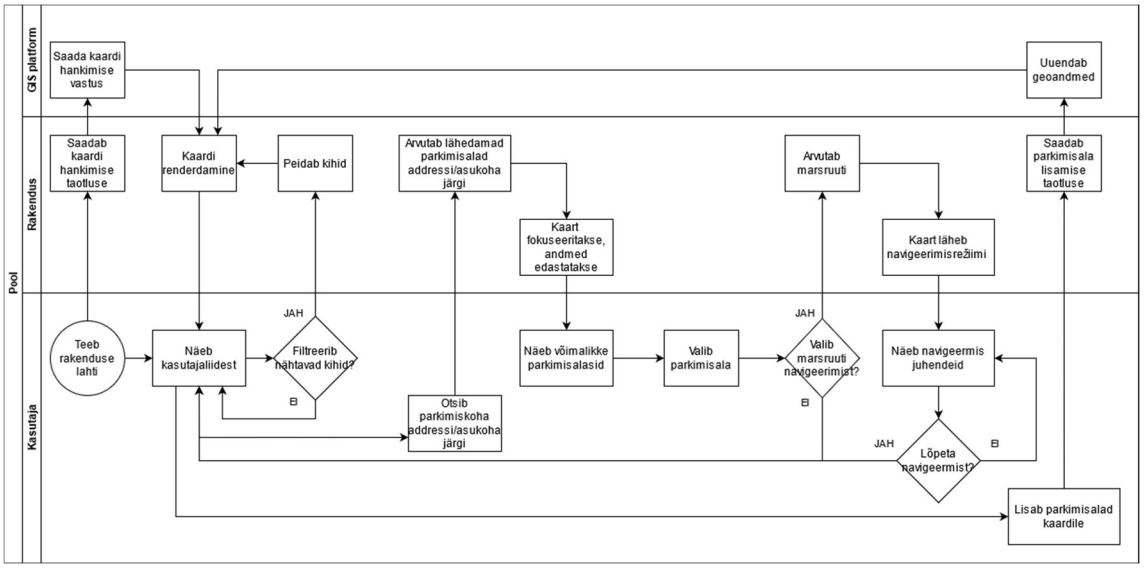
Tabel 2 Kasutajalood

Epic ID	**Kasutajaloo prioriteet**	**Kasutajaloo ID**	**<sellise tüübi> kasutajana**	**Ma tahan <täita mõnda ülesannet>**	**et saaksin <mingi eesmärgi saavutada>**
1	Suur	1	Rakenduse kasutajana	Ma tahan näha parkimisalade kaarti	et saaksin otsida parkimisalasid.
1	Suur	2	Rakenduse kasutajana	Ma tahan näha kaardi legendi	et saaksin kiiremini kaardil navigeerida.
1	Väike	3	Rakenduse kasutajana	Ma tahan filtreerida kaardi legendi elemente	et saaksin näha ainult mind huvitavat infot.
1	Väike	4	Rakenduse kasutajana	Ma tahan muuta kaardi mõõtmeid nuppudega	Ee saaksin rakendust ühe sõrmega kasutada.
2	Suur	5	Rakenduse kasutajana	Ma tahan otsida lähima parkimisala aadressi järgi	et saaksin valida, mis parkimisala kasutada.

2	Väike	6	Rakenduse kasutajana	Ma tahan otsida lähima parkimisala minu asukoha järgi	et saaksin liikvel olles kaardil navigeerida.
2	Väike	7	Rakenduse kasutajana	Ma tahan lisada järjehoidjatesse vähemalt ühe asukoha	et saaksin tulevikus kaardiotsinguks järjehoidjat kasutada.
2	Väike	8	Rakenduse kasutajana	Ma tahan kasutada marsruudi navigeerimist	et saaksin juhised parklasse jõudmiseks.
3	Suur	9	Rakenduse kasutajana	Ma tahan lisada andmeid kaardile	et saaksin tulevikus kaardi andmete täielikkust suurendada.
3	Väike	10	Rakenduse kasutajana	Ma tahan muuta olemasolevad andmeid	et saaksin kaardi andmete täpsust suurendada.
3	Väike	11	Rakenduse kasutajana	Ma tahan eemaldada andmeid kaardilt	et saaksin kaardi andmete õigust suurendada.

Äriprotsessi diagrammi (Joonis 1) järgi on hästi näha, mitte ainult milliseid rakenduse osasid on vaja arendada, vaid ka millises rakenduse arhitektuuri kihis on vaja lisada funktsionaalsus kasutajaloo täitmiseks. Äriprotsessi diagrammis on kolm kihti: Kasutaja, Rakendus ja GIS platvorm. Neid saab hästi seostada MVVM arhitektuuri kihtidega:

- Model vastab GIS platvormile, kuna GIS platvormil hoiab rakenduse töötamiseks vajalikke geoandmeid.
- ViewModelile vastab Rakendusele, kuna selles toimub andmete töötlemine ja arvutamised View jaoks.
- View vastab Kasutajale, kuna selles toimub ainult kasutajaliidese interaktsioon.

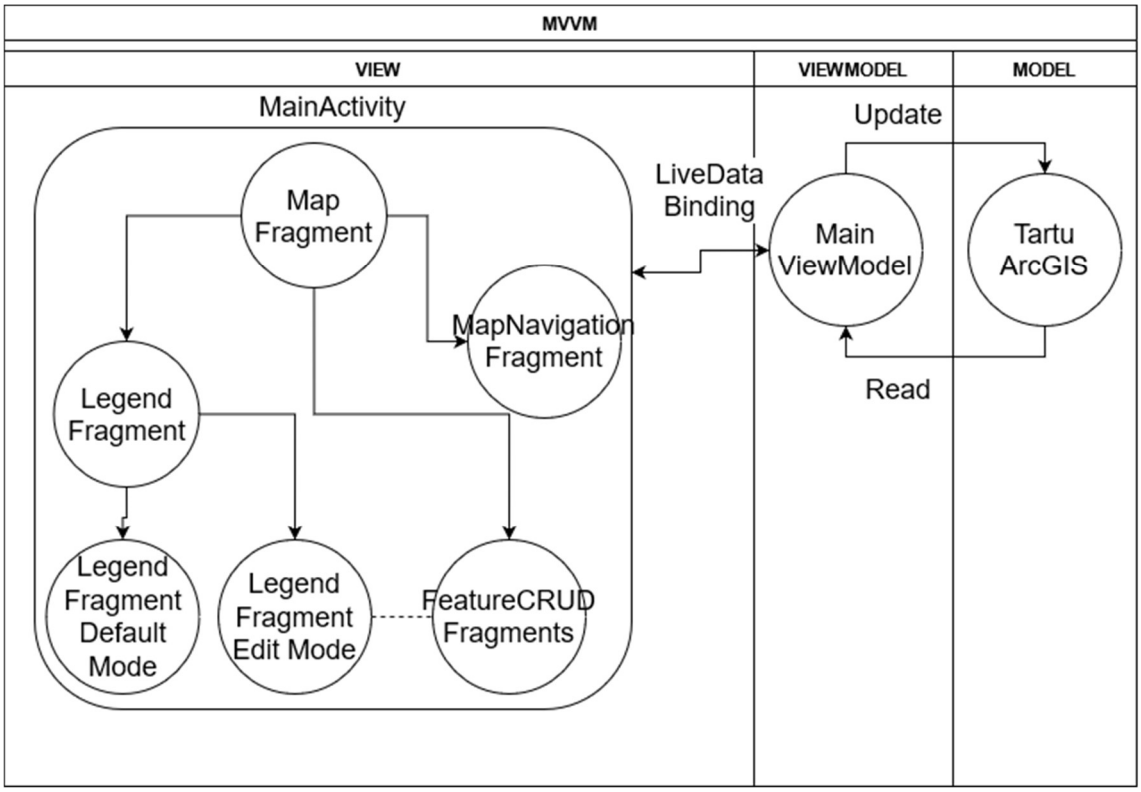


Joonis 1 Äriprotsess

Kuna lõputöö raames arendatava Androidi rakenduse jaoks kasutatakse olemasolevat Tartu ArcGIS platvormi, siis Model kihi arendamine ei ole vajalik, vaid peab kindlaks tegema, et ViewModel ja View kihid vastaksid Model kihile.

4.1 Rakenduse MVVM arhitektuuri rakendamine ja kasutatavad teegid

Rakenduse MVVM arhitektuuri skeemi (Joonis 2) järgi on näha üldine rakenduse ehitus. Tartu ArcGIS platvorm vastutab parkimisalade andmete hoidmise eest. MainViewModel loeb ja uuendab Tartu ArcGIS platvormi andmed ning vahetab andmed MainActivity'ga. MainActivity mängib vaate elutsükli omaniku(*viewLifecycleOwner*) rolli ehk läbi tema toimub kõikide Fragmentide näitamine vastavalt kasutaja tegevustele ning läbi tema kõik Fragmentid saavad liigipäsu MainViewModelile, mis võimaldab ühte ja samade andmete kasutamise kogu rakenduse jaoks.



Joonis 2 Rakenduse MVVM arhitektuuri skeem

4.1.1 Tartu ArcGIS platvorm

Tartu ArcGIS platvorm on saadaval veebiaadressil Tartu Linnavalitsuse GIS portaalis [14] ning peale parkimisalade andmete hoiab selliseid andmeid nagu parkmisautomaatid ja rattaparklad. Erinevat tüüpi andmeid hoitakse erinevates kihtides *(layer*). Selleks, et paremini aru saada, kuidas selle platvormi jaoks Androidi rakendust arendada on vaja vaadata üle need kihid ning kuidas on ehitatud nende kihtide andmetüübid. Vaatamata sellele, et selle lõputöö raames eesmärgiks on arendada rakendust just autode parkimisalade otsimiseks, Tartu ArcGIS platvorm hoiab ka muud kasulike andmed. Täpselt on neli kihti, mis pakuvad kõige kasulikumad andmed:

1. Tasuta parkla kiht hoiab andmed elektriauto laadimiskohtadest, invakohtadest, ajalise piiranguga, bussi ja mootoratta parkimisalade kohta. Need andmed ilmuvad kaardil ikoonidega. Need andmed on kasulikud, kuna neid saab kasutada erilise transpordivahendi juht.

2. Rattahoidiku kiht hoiab andmed rattahoidikutest ehk ratta parkimisohtades, mis on näha kaardipeal ikoonina. Need andmed on ilmselgelt kasulikud jalgratta kasutajatele.

3. Rattaringluse parkla kiht näitab Tartu rattaringluse jalgrattade parkimislalad. Neid on näha samas ikoonina. Vaatamata sellele, et Tartu rattaringluse jaoks on tehtud eraldi rakendus, need andmed võivad minna kasuks näiteks turisti jaoks, kes Tartusse sõidab autoga ning alguses otsib autode parkimisalad ning kaardil võib näha ka rattaringluse parklaid, millega kaasneb huvi selle vastu.

4. Parkimisalad kiht on kõige olulisem ning näitab tasulisi, tööpäeviti tasulisi ja tasuta parkimisalasid kaardil geograafiliste kujudega ehk polügoonidega.

4.1.2 MainViewModel

MainViewModel arendamiseks on vaja kasutada Android Lifecycle teegid [17], mida saab lisada Gradle'sse järgmiste sõltuvuste deklareerimisega:

```
// ViewModel
implementation "androidx.lifecycle:lifecycle-viewmodel-
ktx:$lifecycle_version"
// LiveData
implementation "androidx.lifecycle:lifecycle-livedata-
ktx:$lifecycle_version"
```

Joonis 3 Android Lifecycle teekide sõltuvuste lisamine

ViewModel ja LiveData on MVVM [13] rakendamise jaoks vajalikke klasside implementatsioon, mida pakub kasutamiseks Google.

Samas MainViewModeli kasutamiseks Fragmentides läbi Activity on vaja veel üht sõltuvust:

```
implementation "androidx.fragment:fragment-ktx:1.3.3"
```

Joonis 4 Fragment ktx teegi sõltuvuse lisamine

Põhimõtteliselt küll on võimalik MainViewModeli kasutada Fragmentides ilma seda sõltuvust, kuid selle kasutamisel seda funktsiooni saab rakendada ühe koodi reaga ning vältida võimalikke mälu kaosid vigase implementeerimise puhul.

MainViewModelis olevate andmete kirjeldust võib näha peatükis 4.2.

4.1.3 MainActivity ja Fragmentid

MainActivity vastutab mitme asja vastu, nendeks on:

- vaate elutsükkel;
- andmevahetus MainViewModeli ja Fragmentide vahel;
- vajalike lubade taotlus kasutajalt, näiteks seadme asukoha jälgimine.

Samal ajal põhiliseks Fragmentiks, millest algab vaate elutsükkel on MapFragment, mis vastutab:

- kaardi näitamisest;
- kaardil navigeerimisest;
- Fragmentide ja Dialogide vahel navigeerimisest.

Kaardi näitamise, uurimise, navigeerimise ja uuendamise funktsioonaalsuse rakendamiseks on olemas ArcGIS Androidi rakenduse programmeerimisliides (*ArcGIS Runtime API for Android*), mis kujutab ennast teeki, mis on mõeldud natiivse Androidi ArcGIS kaarte ja teenuseid kasutatava rakenduse loomise jaoks. Teegi dokumentatsioon pakub head ülevaadet teegi võimalustest ning seletusi ja koodijuppe nende rakendamiseks Java ja Kotlin programeerimiskeeltes [18]. Selle teegi kasutamiseks on vaja deklareerida vastav sõltuvus:

```
implementation 'com.esri.arcgisruntime:arcgis-android:100.11.0'
```

Joonis 5 ArcGIS Androidi rakenduse programmeerimisliidese teegi sõltuvuse lisamine

Peale seda selleks, et kasutada ArcGIS Androidi rakenduse programmeerimisliidest on vaja registreerida arendajana ArcGIS veebiportaalil ning taotleda API võti oma ülesannete jaoks [19]. See on tasuta ning sinna on võimalik registreerida Google kontoga. Pärast seda on vaja rakenduses kasutada saadud API võtit andmete taotlemisel:

```
ArcGISRuntimeEnvironment.setApiKey("<API võti>")
val portal = Portal("https://gis.tartulv.ee/portal", false)
val itemId = "752035669d9e42d3be73f9662e1283b4"
val map = ArcGISMap(PortalItem(portal, itemId))
```

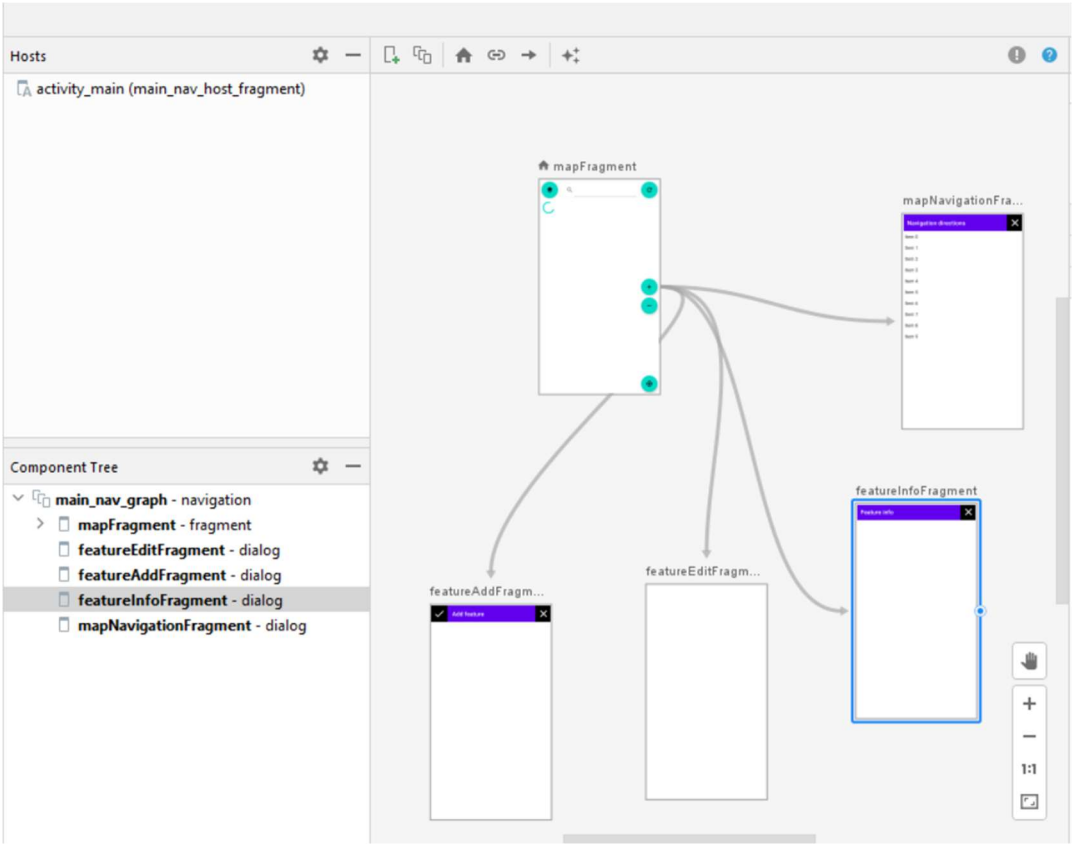
Joonis 6 ArcGIS API võtme seadistamine ja ArcGISMap objekti taotlemine

Fragmentide ja Dialogide vahel navigeerimine toimub navigeerimise graafi teegi kasutamisel [20], mille jaoks on vaja deklareerida järgmised sõltuvused:

```
implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
```

Joonis 7 Android Navigation teekide sõltuvuste lisamine

Selle teegi kasutamisel võib lihtsustada Fragmentide ja Dialogide vahel navigeerimist ning MapFragmenti kood teha tunduvalt puhtam, liigutades osa koodi XML navigeerimise graafi faili sisse.



Joonis 8 Navigeerimise graafi struktuur

Joonis 8 näitab, kuidas visuaalselt näeb välja navigeerimise graaf.

4.2 Agiilse metoodika abil rakenduse arendamine

Agiilse metoodika abil rakenduse arendamine on suunatud aja- ja inimressurside säästvale arendusele. Selle lõputöö raames kasutatavate *epic*ute ja kasutajaloo põhimõtteks on jagada arendaja töö tükkideks [21], mis oleksid väiksemad kui kogu rakendus ning omaksid mingit funktsionaalset väärtust. Kokku selle lõputöö raames on välja toodud kolm *epic*ut:

1. *Epic* ID 1, mis seletab, mis on nõutud rakenduse kaardi uurimise funktsionaalsuse jaoks.

2. *Epic* ID 2, mis seletab, mis on nõutud rakenduse kaardi navigeerimise funktsionaalsuse jaoks.

3. *Epic* ID 3, mis seletab, mis on nõutud rakenduse kaardi uuendamise funktsionaalsuse jaoks.

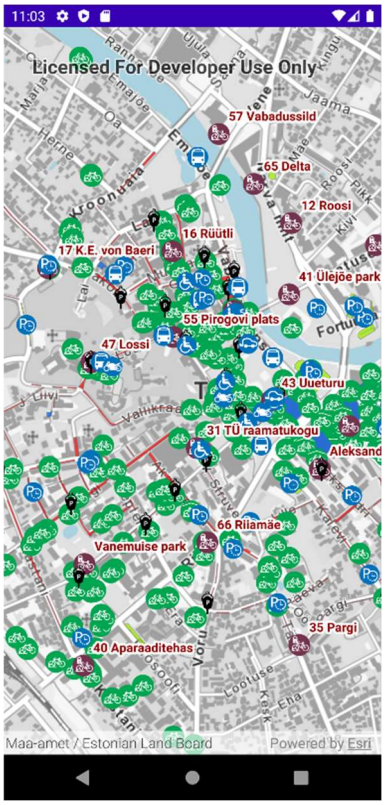
Iga *epic* on jagatud erineva tähtsusega kasutajalugudeks, mis seletavad, millisest funktsionaalsusest rakenduse kasutajale kasu on.

4.2.1 Rakenduse kaardi uurimise funktsionaalsus

Esimene *epic* seletab, mis on nõutud funktsionaalsus kaardi uurimise jaoks ja millist ülesannet tahab täita rakenduse kasutaja kaardi uurimise ajal. Vastavalt esimesele *epic*u kasutajalugudele nõuab rakendus, et oleks võimalik näha kaarti ja selle legendi. Selle funktsionaalsuse realiseerimine on kõige olulisem, kuna kaardi näitamine on ilmselgelt vajalik teise ja kolmanda *epic*u funktsionaalsuse realiseerimiseks.

4.2.1.1 Kaardi näitamine

Esimese *epic*u kaardi näitamise funktsionaalsuse rakendamiseks kasutades ArcGIS Androidi rakenduse programmeerimisliidest on vaja kasutada MapView vaate klassi ja ArcGISMap andmete klassi. [19]



Joonis 9 MapView puhas kaart

MapView parameeter *map*, mis võtab sisse ArcGISMap klass objekti vastutab selle eest, millist kaardi ta näitab, ning kujutab endast puhast kaarti, mille peal ei ole legendi ega erilist navigeermis võimalusi (Joonis 9). ArcGISMap objekti hankimiseks on vaja kasutada ArcGIS arendaja API võtit ning hankida Tartu ArcGIS platvormist vajalik kaart. Vastavalt MVVM arhitketuurile MainViewModelis on loodud *mapLiveData* objekt, mille sisse hangitakse Tartu ArcGIS platvormist andmed ning mida kasutab MapFragmentis asuv MapView kaardi näitamiseks:

```
val mapLiveData: LiveData<ArcGISMap>
```

Joonis 10 ArcGisMap objekti hoidev LiveData objekt

4.2.1.2 Kaardi legendi näitamine ja filtreerimine

ArcGISMap objekti sees on hoitud kõik kaardil nähtavad kihid, näiteks tasulised ja tasuta parkimisalad. Selleks, et saada nende kihtide kaardilegend, on vaja itereerida läbi kõiki kihte ja hankida LegendInfo objektide list, mille sees on legendi elementide nimed ning tähised:

```
arcGisMap.operationalLayers.forEach { operationalLayer ->
       val layerLegend: ListenableFuture<List<LegendInfo>> =
operationalLayer.fetchLegendInfosAsync()
       layerLegend.addDoneListener {
              layerLegend layerLegend.get())                }
}
```

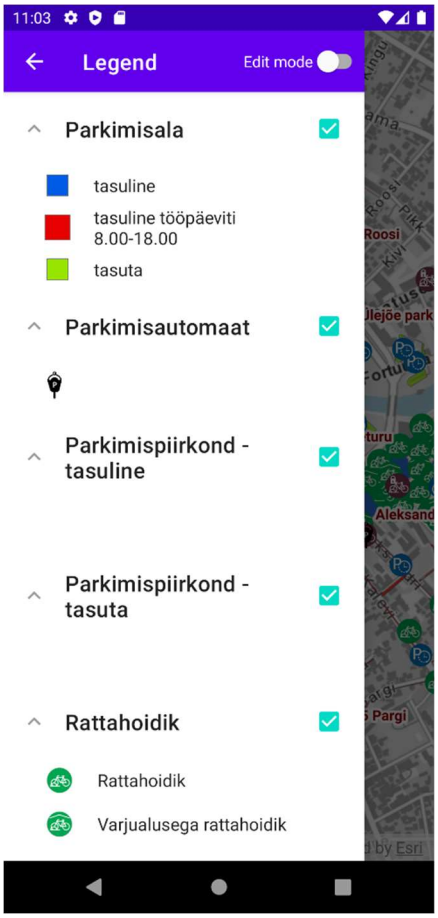
Joonis 11 Kaardi legendi päring ArcGIS platvormilt

Saadud ArcGISMap objekt ja LegendInfo objektid on mõistlik salvestada ViewModelis LiveData objektide sees ning hankida need otse rakenduse käivitamisel. *legendLayerAndInfoLiveData* objekt kasutakse LegendFragment vaates legendi loomiseks.

```
val legendLayerAndInfoLiveData: MutableLiveData<MutableMap<Layer,
List<LegendInfo>>>
```

Joonis 12 Kaardi legendi infot hoidev LiveData objekt

LegendFragment kujutab ennast niinimetatud Navigation draweri, mis Material disani juhtnööride järgi [22] kasutatakse erinevate Fragmentide vahel navigeerimiseks (Joonis 13). Vaatamata sellele autori arust Navigation drawerit on mõistlik kasutada legendi hoidmiseks, kuna legendi vaatamine ja filtreerimine on tihti nagu tavalises rakenduses navigeerimine.



Joonis 13 MapView kihtide legend

Kihtide filtreerimist on võimalik samal kohal rakendada kasutades märkeruute.

Legendi elementide näitamiseks on kõige mõistlikum kasutada RecyclerView vaate klassi [23]. See klaas võimaldab dünaamilise listi elementidest luua järjendi legendi elementidest. Selline lähenemine võimaldab ka näidata erinevat hulka legendi elemente ehk see teeb rakenduse hästi laiendatavaks, sest kui Tartu ArcGIS platvormi kaardile lisatakse uued legendi elemendid, siis ei ole vaja mingit rakenduse uuendamist teha.

4.2.1.3 Kaardi mõõtmete muutumine

MapView võimaldab kaardil liikumise ja mõõtmete muutumise žestidega, kuid žestide kasutamine ei pruugi alati olla mugav või võimalik. Kaardi mõõtmete muutumisest on tehtud kaks nuppu, mis võimaldavad kiirelt ja ühe sõrmega sisse ja välja suumida.

4.2.2 Rakenduse kaardi navigeerimise funktsionaalsus

Teine *epic* seletab, mis on nõutud funktsionaalsus kaardi navigeerimise jaoks ja millist ülesannet tahab täita rakenduse kasutaja kaardi navigeerimise ajal. Vastavalt teisele *epic*u kasutajalugudele nõuab rakendus, et oleks võimalik navigeerida aadressi ja asukoha järgi.

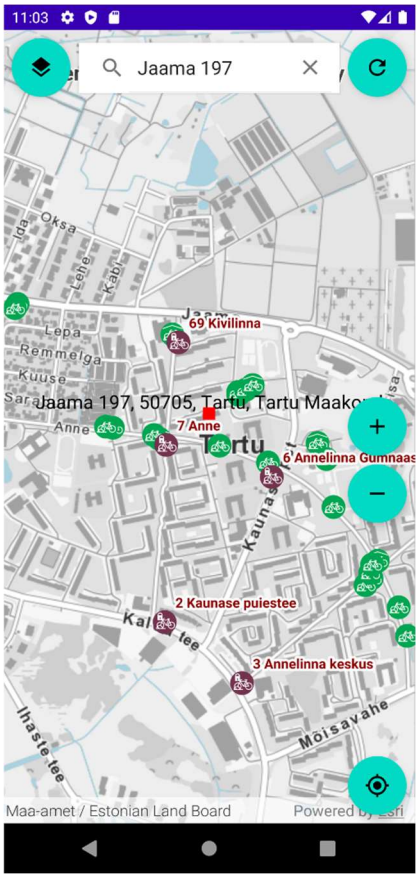
4.2.2.1 Aadressi otsimine rakenduses

Aadressi otsimist ArcGIS MapView kaardil on võimalik rakendada kasutades LocatorTask klassi [24]:

```
private val locatorTask =
LocatorTask("https://geocode.arcgis.com/arcgis/rest/services/World/Geocode
Server")
```

Joonis 14 LocatorTask objekti loomine

LocatorTask kasutab ArcGISe tasuta teenust, mis muudab sõnalised aadressid geokodeeritud andmeteks ehk näiteks koordinaatpunktiks. Selle kasutamiseks on kõige mõistlikum kasutada Androidi SearchView vaadet, mis kujutab endast otsinguriba.



Joonis 15 Jaama 197. aadressi otsimine rakenduses

Sellisel rakendamisel kasutaja saab kasutada otsinguriba aadressi sisestamiseks, ning LocatorTask muudab selle aadressi kaardipunktiks (Joonis 15). Kaardipunkti saab kaardile joonistada kasutades GraphicsOverlay klassi, mis on põhimõtteliselt MapView graafika renderdusmootor. Kuna LocatorTask otsib aadresse kogu maailmast, siis Tartu jaoks võiks automaatselt, kasutajale nähtamatult, lisada juba sõna "Tartu" LocatorTaski veebipäringusse.

4.2.2.2 Nutiseadme asukoha kasutamine

Nutiseadme asukoha navigeerimise kasutamise võimaluse loomiseks on põhimõtteliselt vaja luua nupp kahe režiimi vahetamiseks, kuna MapView pakkub väga mugavat programeerimisliidest kaardi navigatsiooni ja uurimise režiimi vahetamiseks [25].

4.2.2.3 Marsruudi navigeerimine

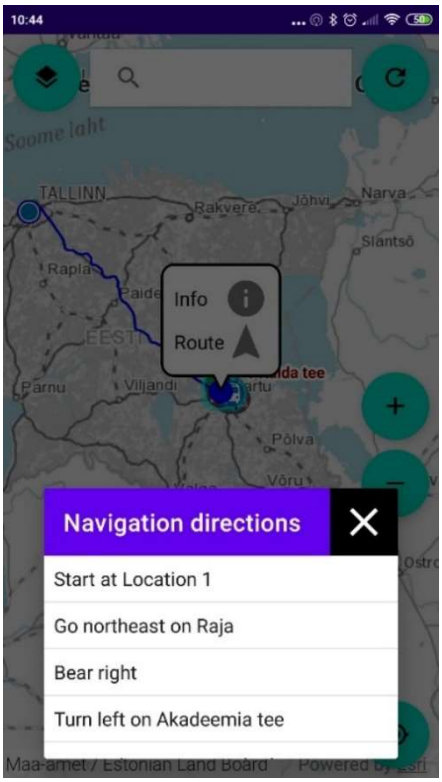
Marsruudi navigeermise realiseerimiseks on võimalik kasutada RouteTask klassi [26]:

```
val routeTask = RouteTask(this,"https://route-
api.arcgis.com/arcgis/rest/services/World/Route/NAServer/Route
_World"         )
```

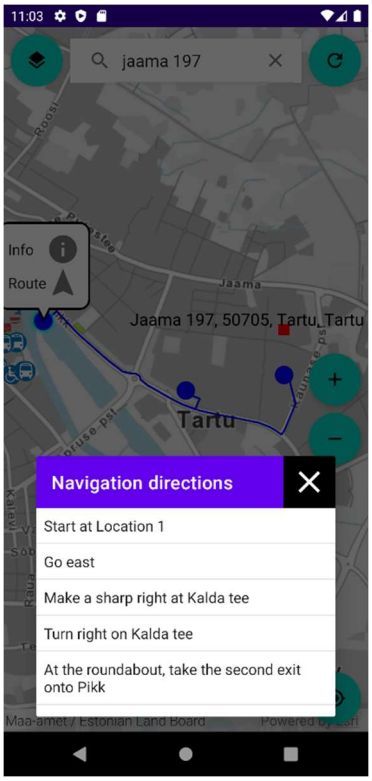
Joonis 16 RouteTask objekti loomine

, mis võimaldab kasutada ArcGIS navigeerimise tasuta teenust. Selle kasutamisel on võimalik valida kaks või rohkem punkti kaardi peal, hankida vastav marsruut, mille rakendus kuvab joonena kaardi peal, kui ka näidata kasutajale juhised valitud puntide vahel. Rakenduses on loodud sellega kaks marsruudi navigeerimise võimalust:

1. kaardi keskmisest punktist kaardiobjektini, näiteks parkimisalani;
2. nutiseade asukohast kaardiobjektini (Joonis 17).



Joonis 17 Navigeerimine asukoha järgi



Joonis 18 Navigeerimine aadressi järgi

Marsruudi navigeerimise kasutamiseks on vaja vajutada huvi pakkuva kaardiobjekti peale ning väikesest menüüst valida marsruudi navigeerimine. Tänu sellele, et aadressite

otsimise puhul kaart tsentreerib otsival aadressil, on võimalik ka suure täpsusega leida marsruut mingist aadressist mingi kaardiobjektini (Joonis 18).

4.2.2.4 Järjehoidjasse andmete lisamine

Järjehoidjasee andmete lisamiseks on kasutatud ArcGIS Androidi rakenduse programmeerimise liidese UI komponenti Bookmark. See võimaldab mingi objekti andmete salvestamise ja kasutamise tulevikus. [27]

4.2.3 Rakenduse kaardi uuendamise funktsionaalsus

Rakenduse kaardi uuendamise funktsionaalsuse loomiseks on tehtud otsus luua eraldi kaardi redigeerimise režiimi ning redigeerimise režiimi kasutamisel on võimalik nähtavaks teha ainult üks kiht. See on põhjustatud nii rakenduse kasutamise mugavdamisega kui ka rakenduse arendamise lihtsustamisega. Selle režiimi saab valida LegendFragmentis ning selle valimine muudab natuke LegendFragmenti vaatet. Redigeerimise režiimi valimisel MainViewModel saab teada, et rakenduse režiim on muudetud ning annab sellest teada ka MapFragmentile. MapFragment vastavalt sellele muutub mitmetel viisidel:

1. marsruudi navigeerimine ei ole enam võimalik;
2. kaardi peale ilmub märk, mis näitab kaardi keskmist kohta;
3. ilmuvad uued nupud;
4. ei ole võimalik lihtsalt näha kaardi objekti andmeid, selle asemel nüüd on kolm eraldi Fragmenti, mis võimaldavad lisada uued objektid valitud kihi tüüpi, muuta olemas olevate objektide andmeid ja kustuda andmeid kaardilt.

Viimase muutuse puhul kõik tegevused saadetakse ArcGIS platvormi serverile ning andmed uuendatakse ka seal.

Uue objekti lisamine toimub nupu vajutamisel ning andmed lisatakse kaardi keskmisele punktile [28]. Objekti lisamisel on võimalik ise valida kõik parameetrid peale andmebaasi ID.

Olemasoleva objekti andmete muutumine toimub selle objekti peale vajutamisel ning menüüst andmete muutmise valimisel [29]. Sama moodi nagu uue objekti lisamisel on võimalik muuta kõiki parameetreid peale andmebaasi ID.

Objekti kustutamine toimub objekti peale vajutamisel ning menüüst objekti kustutamise valimisel. [30]

Kõik uued Fragmentid on loodud samamoodi nagu LegendFragment ehk kasutades RecyclerView klassi. Nagu LegendFragmneti puhul, võimaldab see ükskõik mitme välja näitamist vastavalt ArcGIS platvormi andmebaasi andmetele [23]. Ehk samamoodi andmete laiendamisel rakenduse laiendamist ei ole otseselt vaja. Samas samamoodi kasutavad need Fragmentid MainViewModelis hoitud andmeid. Ehk näiteks kaardi peal objekti valimisel saadakse selle objekti andmed MainViewModelisse ning vastavalt sellele hakkab Fragment ehitama oma vaateid.

Kahjuks Tartu ArcGIS platvormi näitel ei ole võimalik kontrollida seda funktsionaalsust, kuna Tartu ArcGIS platvorm on hetkel suletud redigeerimiseks ning selle jaoks on vaja eraldi API võtit. Kuid isetehtud avaliku platvormi näitel kõik funktsionaalsus töötab ja kuna rakendus on arendatud võttes kaasa võimaliku platvormi laiendamist, tähendab see, et rakendus peaks toimima ka Tartu ArcGIS platvormi puhul. Selle jaoks, et edasi testida rakendust on vaja läbirääkida seda Tartu Linnavalitsusega.

4.2.4 Rakenduse lõplik arhitektuur

Tabel 3 näitab, kuidas näeb rakenduse lõpplik arhitektuur ja kuidas on seotud LiveData objektid MainViewModeli sees ja vaated.

Tabel 3 LiveData objektide kasutus vaadetes

MainViewModel	**View**	**Ülesanne**
mapLiveData - ArcGISMap	MapFragment	Kaardi näitamine
legendLiveData - Map<FeatureLayer, List<LegendInfo>>	LegendFragment	Legendi näitamine
isEditModeLiveData - Boolean	MapFragment, LegendFragment	Redigeerimisrežiimi sisse/välja lülitamine
selectedLayerToEditLiveData - FeatureLayer	MapFragment, FeatureAddFragment	Geoobjekti lisamine ja eemaldamine

selectedGeometryLiveData - Geometry	FeatureAddFragment	Geoobjekti lisamine
selectedFeatureLiveData - ArcGISFeature	FeatureEditFragment, FeatureInfoFragment	Geoobjekti andmete vaatamine ja redigeerimine
mapNavigationDirections - List<String>	MapNavigationFragment	Navigeerimise juhiste näitamine

MVVM arhitektuuri rakendamine võimaldas luua süsteemi, kus mitmed Fragmentid saavad kasutada samu andmeid, mille pärast tõenäoliselt rakenduse jõudlus suureneb. Lõplik arhitektuur meenutab MVC ja MVVM kombinatsiooni. See võib olla põhjustatud ArcGIS Androidi rakenduse programmeerimise liidese erilisusega. Enamus navigeerimisega seotud meetodeid ei kasuta MVVM arhitektuuri, kuna nende rakendamine on otseselt seotud MapView vaatega ja eraldi LiveData andmeobjekti nende jaoks ei ole mõistlik luua. Vaatamata sellele, olemasolev arhitektuur võimaldab lahendada kõik lõputöös püstitatud probleemid.

4.2.5 Rakenduse levitamine

Androidi rakenduse levitamiseks on olemas kaks võimalust: levitada seda läbi Google Play Store ja läbi Google Firebase [31]. Kuna Google Play Store'is arendajana registreerimiseks on vaja kõige pealt maksta siis ilmselgelt oli valitud Google Firebase. Peale seda Google Firebase võimaldab jagada rakendust vaid teatud inimestele, mis suurendab turvalisust. Google Firebase'i kasutamiseks on vaja lihtsalt registreerida ennast ja rakendus, mida sa tahad levitada. Google Firebase loob selle jaoks JSON faili, mis on vaja oma Androidi rakenduse projekti sisse lisada ning lisada paar teegi sõltuvust. Pärast seda on vaja lihtsalt ehitada rakenduse APK faili ning üleslaadida see Firebase platvormile, lisades inimeste, kellele on vaja rakendust levitada, meilid. Kuna rakendus on alles jõudnud MVP olekuni see on tõesti kõige odavam, lihtsam ja turvalisem viis rakendust levitada.

5 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua töötav Androidi rakendus agiilse metoodika abil, mis võimaldab parkimisalade kaardi uurimist, navigeerimist ja uuendamist, see omalt poolt lahendab parkimisalade otsimise probleemi Tartus, mida ei lahenda teised turul olevad Androidi rakendused.

Käesoleva töö peamine tulemus on töötav Androidi rakendus, mis vastab autori välja toodud rakenduse nõuetele. Vastavalt nõuetele on see rakendus ka laiendatav ning selle lähtekood võib olla kasutatud mingi muu eesmärgi kasutamiseks, näiteks postiljonide töö lihtsustamiseks mõeldud rakenduse loomiseks.

Antud töö kirjeldab ja seletab, kuidas võib toimuda Androidi rakenduse agiilne arendamine, kuidas ArcGIS platvormi saab kasutada Androidi rakenduses ja kuidas realiseerida mõningad Androidi ArcGIS Runtime API võimalused. Lisaks sellele, analüüsib töö lühidalt teisi võimalusi ja valikud, mida saab kasutada

Lõpude lõpuks rakendus on hea MVP, mis omab täpselt nii palju väärtust, kui on vaja, et otsustada, kas selle edasi arendamine on mõistlik. Kuna Eesti avaandmete platvorm areneb hetkel kiiresti ja ArcGIS platvorm on hea geoandmete kogumiseks, siis autor usub, et selle lõputöö raames loodud rakendus leiab kasutust. Selle rakenduse autor plaanib esitada ka Tartu Linnavalitsusele sooviavalduse pakkuda koostööd selle rakenduse edasises arendamises, muutmises ja testimises.

Kasutatud kirjandus

[1] H. K. R. Y. Evgeny Aleksandrov, „The Impact of COVID-19 on U.S. Food Delivery Services | by Ryan Young | Towards Data Science,“ 19 Jaanuar 2021. [Võrgumaterjal]. Available: https://towardsdatascience.com/the-impact-of-covid-19-on-food-delivery-services-in-the-u-s-47eae04655c8. [Kasutatud 29 Aprill 2021].

[2] K. Kera, “Tasuline parkimine Tartus,” Tartu Linnavalitsus, 1 September 2020. [Online]. Available: https://gis.tartulv.ee/portal/apps/webappviewer/index.html?id=3a7b615f61f24677b3cf076c69b1f805&mobileBreakPoint=3.. [Accessed 17 Mai 2021].

[3] Google, “Google Maps - Navigate & Explore - Apps on Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.maps. [Accessed 17 Mai 2021].

[4] Waze, “Waze - GPS, Maps, Traffic Alerts & Live Navigation - Apps on Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=com.waze. [Accessed 17 Mai 2021].

[5] Sygic, “Sygic GPS Navigation & Offline Maps - Apps on Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=com.sygic.aura. [Accessed 17 Mai 2021].

[6] OPENGIS.ch, “QField for QGIS - Apps on Google Play,” [Online]. Available: QField for QGIS. [Accessed 17 Mai 2021].

[7] Statcounter, “Mobile Operating System Market Share Worldwide,” [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed 17 Mai 2021].

[8] JetBrains, “Kotlin docs | Kotlin,” [Online]. Available: https://kotlinlang.org/docs/home.html. [Accessed 17 Mai 2021].

[9] QGIS, “Discover QGIS,” [Online]. Available: https://qgis.org/en/site/about/index.html. [Accessed 17 Mai 2021].

[10] Esri, “ArcGIS Online | Cloud-Based GIS Mapping Software,” [Online]. Available: https://www.esri.com/en-us/arcgis/products/arcgis-online/overview. [Accessed 17 Mai 2021].

[11] C. S. K. M. Bill Phillips, Android Programming: The Big Nerd Ranch Guide, 4th Edition, Big Nerd Ranch Guides, 2019.

[12] B. Pandey, “Model View Presenter(MVP) in Android with a simple demo project,” 6 Detsember 2017. [Online]. Available: https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42. [Accessed 17 Mai 2021].

[13] Google, “Guide to app architecture | Android Developers,” [Online]. Available: https://developer.android.com/jetpack/guide. [Accessed 17 Mai 2021].

[14] E. Maxwell, “MVC vs. MVP vs. MVVM on Android,” 26 Jaanuar 2017. [Online]. Available: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/. [Accessed 17 Mai 2021].

[15] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Pearson, 2008.

[16] K. Kera, “Parkimisalad Tartu linnas,” Tartu Linnavalitsus, 1 September 2020. [Online]. Available: https://gis.tartulv.ee/portal/home/item.html?id=752035669d9e42d3be73f9662e1283b4 . [Accessed 17 Mai 2021].

[17] Google, “Lifecycle | Android Developers,” [Online]. Available: https://developer.android.com/jetpack/androidx/releases/lifecycle. [Accessed 17 Mai 2021].

[18] Esri, „ArcGIS Runtime API for Android,“ 2021. [Võrgumaterjal]. Available: https://developers.arcgis.com/android/. [Kasutatud 29 Aprill 2021].

[19] Esri, “Display a map | ArcGIS Runtime API for Android,” 2021. [Online]. Available: https://developers.arcgis.com/android/maps-2d/tutorials/display-a-map/. [Accessed 17 Mai 2021].

[20] Google, “Get started with the Navigation component | Android Developers,” [Online]. Available: https://developer.android.com/guide/navigation/navigation-getting-started. [Accessed 17 Mai 2021].

[21] M. Rehkopf, “Epics | Atlassian,” Atlassian, [Online]. Available: https://www.atlassian.com/agile/project-management/epics. [Accessed 17 Mai 2021].

[22] Google, “Navigation drawer - Material Design,” [Online]. Available: https://material.io/components/navigation-drawer. [Accessed 17 Mai 2021].

[23] Google, “Create dynamic lists with RecyclerView | Android Developers,” [Online]. Available: https://developer.android.com/guide/topics/ui/layout/recyclerview. [Accessed 17 Mai 2021].

[24] Esri, “Search for an address | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/geocode-and-search/tutorials/search-for-an-address/. [Accessed 17 Mai 2021].

[25] Esri, “Display device location | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/kotlin/sample-code/display-device-location/. [Accessed 17 Mai 2021].

[26] Esri, “Find a route and directions | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/route-and-directions/tutorials/find-a-route-and-directions/. [Accessed 17 Mai 2021].

[27] Esri, “UI components | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/ui-components/. [Accessed 17 Mai 2021].

[28] Esri, “Add features (Feature Service) | ArcGIS Runtime API for Android,” 2021. [Online]. Available: https://developers.arcgis.com/android/java/sample-code/add-features-feature-service/. [Accessed 17 Mai 2021].

[29] Esri, “Feature layer update attributes | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/java/sample-code/feature-layer-update-attributes/. [Accessed 17 Mai 2021].

[30] Esri, “Delete features (feature service) | ArcGIS Runtime API for Android,” [Online]. Available: https://developers.arcgis.com/android/java/sample-code/delete-features-feature-service/. [Accessed 17 Mai 2021].

[31] Google, “Documentation | Firebase,” [Online]. Available: https://firebase.google.com/docs. [Accessed 17 Mai 2021].

[32] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[33] WHO/HQ/WHE/HIM/MAP (GIS team), „Comparison of Geographic Information Systems (GIS),“ 12 Märts 2018. [Võrgumaterjal]. Available: https://www.who.int/health-cluster/resources/publications/OpenSourceGISComparison.pdf. [Kasutatud 29 Aprill 2021].

[34] R. C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, First Edition, Pearson, 2017.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Andrei Gužovski

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Parkimisalade otsimise platvormi arendus Tartu näitel", mille juhendaja on Aleksei Talisainen
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.