TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Eerik Kompus 120820IABB

ÜRITUSTE LEIDMISEKS NING SELLEGA KAASNEVA SÜSTEEMI HALDAMISEKS MÕELDUD ANDROIDI RAKENDUSED KIFT PLATVORMI NÄITEL

bakalaureusetöö

Juhendaja: Roger Kerse

lektor

Tallinn 2017

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: Eerik Kompus

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua kaks Androidi rakendust. Esiteks rakendus, milles kasutajal on võimalik leida enda ümber toimuvaid üritusi ning teiseks rakendus, kust saab hallata kogu ürituste süsteemi.

Autor analüüsib bakalaureusetöös juba olemasolevaid lahendusi ürituste leidmiseks, tutvustab Kift platvormi toimimise loogikat ning uurib erinevaid tehnoloogiaid Androidi rakenduste valmistamiseks.

Töö tulemusena valmisid kaks Androidi rakendust. *Back office* rakendus kogu süsteemi haldamiseks, mis on mõeldud Kift projektiga seotud inimestele ning *client* rakendus, kust saab kiirelt ja mugavalt leida informatsiooni ümbruskonnas toimuvate ürituste kohta.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 4 peatükki, 24 joonist ning 2 tabelit.

Abstract

The purpose of this thesis is to create two Android applications. Firstly an application where a user can find events that are happening nearby and secondly an application for managing all events and the system behind it.

Author analyses solutions that are already on the market, introduces how the Kift platform actually works and investigates different technologies for developing an Android application.

As a result of this thesis two Android applications were made. A back office application for managing the system itself was developed for the people who are working in Kift project and secondly a client application was created where user can easily find events that are happening nearby.

The thesis is in Estonian and contains 48 pages of text, 4 chapters, 24 figures and 2 tables.

Lühendite ja mõistete sõnastik

Android *toast*	Androidis kasutusel olev teavitussüsteem, millega saab kasutajale kuvada lühiajaliselt teateid. [46]
API	***Application programming interface*** Rakenduseliides on reeglistik olemasoleva valmisprogrammiga suhtlemiseks. See võimaldab kirjutada lisaprogramme, mis laiendavad programmi funktsionaalsust. [41]
AsyncTask	**Asünkroone tegevus** Tegevus, mis on võimeline taustal oma ülesande täitma ning ei hoia kasutaja tegevust kinni.
Back office rakendus	**Sisuhaldussüsteemi rakendus** Rakendus, mis on mõeldud süsteemi haldamiseks.
Boiler-plate kood	Koodi osad, mis korduvad samal kujul ühes koodis.
Client rakendus	**Kasutaja rakendus** Rakendus, mis on mõeldud üldisele rahvale.
Cross-platform	**Multiplatvorm** Teenus või lahendus, mis on võimeline töötama korraga mitmel erineval platvormil.
FAB	***Floating Action Button*** Android 5.0-s tutvustatud navigeerimislahendus, mis lisab ekraanile hõljuva nupu, mis ei võta palju ruumi, kuid annab võimaluse vaatesse lisada funktsionaalsust.
Facebook Graph API	Facebooki poolt loodud API, mille abil saab küsida infot Facebookis olevate asjade kohta ning lisada infot Facebooki. [44]
FCM	***Firebase Cloud Messaging*** FCM on Google poolt loodud lahendus läbi mille saab saata sõnumeid ja teateid Androidi, iOSi ja veebirakendustesse. [31]
Google Play Store	Google poolt loodud ametlik teenus, kus kasutaja saab alla laadida ning hallata kõiki rakendusi, mis on loodud Androidi seadmetele ning avaldatud läbi Google. [40]
Hamburger menu	Androidi disainilahendus, kus kogu navigeerimispaneel on viidud ekraanilt välja ning selle nägemiseks on vaja vajutada ekraani ülaosas olevat hamburgeri kujulist nuppu või tõmmata menüü ekraani vasaku ääre tagant välja. [43]
JSON	***JavaScript Object Notation*** JSON on lihtsustatud andmevahetusvorming, mis on tekstiformaadis ning sõltumatu programmeerimiskeeltest. JSON-st on saanud kõige populaarsem viis informatsiooni edastamiseks. [45]

library	**teek** Kollektsioon juba implementeeritud funktsionaalsust, mis on mõeldud korduvkasutamiseks erinevates programmides.
MVP	***Model-View-Presenter*** Androidis tuntud arhitektuuri lahendus, kus kasutajale kuvatav vaade ja taustal toimuvad asünkroonsed tegevused on üksteisest eraldatud. [25]
Responsive design	**Reageeriv disain** Disainilahendus, kus leht on võimeline disaini kohendama vastavalt ekraani suurusele.
Ripple effect	**Sulina effekt** Android 5.0-s tutvustatud puute animatsioon, mis tekitab effekti nagu midagi oleks vette kukkunud ja sellest on tekkinud lained. [42]
UI disain	***User Interface* disain** UI disainiks nimetatakse rakenduse või veebilehe välimust.
UX disain	***User Experience* disain** UX disainiks nimetatakse kasutajakogemuse disainimist. Mille alla kuulub kõik, mis muudab kasutajakogemuse võimalikult lihtsaks, arusaadavaks ning nauditavaks.

Sisukord

Jooniste nimekiri

Tabelite nimekiri

Sissejuhatus

Bakalaureusetöö eesmärgiks seadis autor ürituste leidmiseks ning selle süsteemi haldamiseks mõeldud Androidi rakenduste loomise. Selle saavutamiseks analüüsis autor olemasolevaid lahendusi ning võttis arvesse nende positiivseid ja negatiivseid külgi. Süsteemi haldamiseks mõeldud rakendus on mõeldud projektiga seotud inimestele ning *client* rakendus on eelkõige suunatud inimestele, kes sooviksid aktiivselt oma vaba aega veeta, kuid ei leia tihti selleks väljundit.

Bakalaureusetöö esimeses osas tutvustab autor ürituste leidmisega seonduvat probleemi ning analüüsib olemasolevaid lahendusi, mis pakuvad sarnast teenust nii veebis kui ka nutiseadmetes.

Töö praktilises osas toob autor välja Androidi rakenduste disainimisel enim levinud probleemi, tutvustab Kift platvormi töötamise põhimõtteid ning nimetab mõlemale bakalaureusetöös valmivale rakendusele esitatud nõuded. Samuti analüüsib autor töö tegemiseks valitud programmeerimiskeelt, arhitektuurilist lahendust ning vahendeid, mida töö käigus kasutati.

Viimases osas tutvustab autor mõlemat valminud teenust ning võrdleb *client* rakendust juba olemasolevate lahendustega. Lisaks analüüsib autor rakendustele seatud nõuetele vastavust ning lõpetuseks arutleb edasiarendamise suundade üle.

1. Ürituste leidmise võimalused

Antud peatükis analüüsib autor ürituste leidmisega seotud probleeme ning uurib lähemalt Eestis olemasolevaid ürituste avastamise teenuseid.

1.1 Taust ja probleem

Tänapäeval on esile kerkinud uut laadi probleem: kui vanasti oli inimestel infopuudus, siis nüüd on see asendunud liigse info olemasoluga. On tekkinud probleem, kus kasutajale kuvatakse liiga palju infot, mille seast endale vajaliku teabe leidmine on tihti väga keeruline. [1]

Natuke väiksemas mahus, kuid sarnane probleem on tekkinud ka Facebookis ja täpsemalt üritustega. Facebookis on tohutult erinevaid üritusi ning endale meelepärase leidmine võib tihti olla raske või isegi võimatu. Juba pikemat aega on räägitud, et oleks vaja lahendust, mis annaks lihtsa ja selge ülevaate üritustest ning aitaks inimestel leida just seda üritust, mida ta otsib. [2]

Ürituste leidmiseks olemasolevaid lahendusi on Eestis mitmeid, kuid enamik neist pakuvad kas piiratud funktsionaalsust või on hädas eelmainitud liigse info üleküllusega. *Google Play Store*-st ei leidnud autor ühtegi ürituste leidmise rakendust, mis oleks mõeldud Eesti turule. Autor leidis küll 3 rakendust, mis pakuvad sarnast teenust, kuid ükski ei suutnud kuvada Eestis toimuvaid üritusi. [3, 4, 5]

Tänapäeva mobiilses maailmas on inimeste jaoks esmatähtis kiirus ja mugavus, kuidas info kasutajani jõuab. Seetõttu on oluline, et info ürituste toimumise kohta saadaks kätte võimalikult lihtsalt ja murevabalt.

1.2 Olemasolevad lahendused ürituste leidmiseks Eesti turul

Järgnevates alapeatükkides analüüsib autor lähemalt Eestis kasutusel olevaid ürituste leidmise teenuseid ning veebilehti ja nende lahendusi Androidi platvormil.

1.2.1 Facebook

Facebook on platvorm, milles esinevat probleemi käesolevas bakalaureusetöös lahendada üritatakse. Tegemist on täieliku sotsiaalvõrgustikuga, mis hõlmab endas näiteks inimeste ja kohtade profiile, gruppe, sõnumeid ning suhtlust, uudiste *feedi*, üritusi, tavalisi- ja videokõnesid, mänge, osta ja müü gruppe, *live feedi* ning palju muud. [7]

Facebook haldab suurt kogust infot, et sellist info üleküllust natukenegi hajutada kolis ettevõte oma sõnumite ja suhtluse osa 2011. aastal eraldiseisvasse rakendusse - Messenger. Sarnast sammu on ootatud pikalt ka ürituste osale ning 2016. aasta lõpus tutvustas Facebook uut rakendust nimega ‘Events from Facebook’. [2, 8, 9]

Autoril ei olnud võimalik tutvuda lähemalt Events rakendusega, sest see ei ole kättesaadav Eestis. Täpsustavaks infoks lisab autor, et *Google Play Store*-i rakenduse üleslaadimisel saab rakenduse lisaja määrata piirangu, mis riikides see kättesaadav on. Järgnevalt toob autor välja põhilisemad punktid rakenduse kohta, mis selgusid uuritud artiklitest. [8, 9]

- Rakendus on otseselt seotud kasutaja Facebookiga. Seega esimene nõue rakenduse kasutamiseks on Facebooki olemasolu. [9]
- Rakendus kuvab kõiki Facebooki üritusi ehk jätkuvalt ei ole teada, kuidas üritusi filtreeritakse ning kas see muudab nende leidmise lihtsamaks. [9]
- Rakendus ei ole Eestis kasutatav. [10]

Kuigi autor ei saanud teha täielikku analüüsi Facebook Events rakendusele, saab rakenduse valmimisest siiski järeldada, et ka Facebook mõistab, et ürituste haldamine ja endale kõige sobivama leidmine on jätkuvalt probleemiks ning head lahendust veel leitud ei ole.

1.2.2 Täna Tallinn

Täna Tallinn on enamjaolt Facebookis tegutsev lehekülg. Neil on ka veebilehekülg, millel kuvatakse ainult kindlate kohtade tänaseid üritusi ning täpsema info saamiseks ürituste kohta suunatakse edasi Täna Tallinna Facebooki lehele. [11]

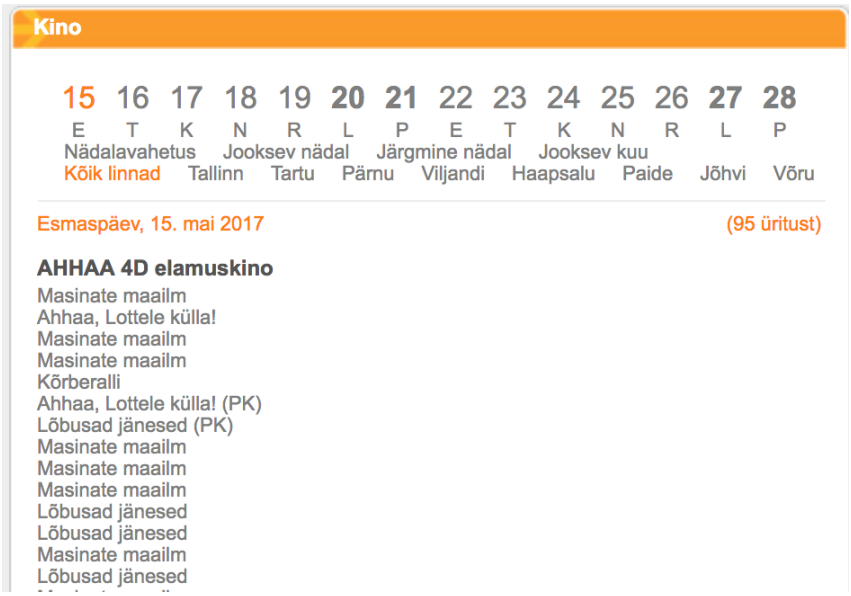
Täna Tallinna Facebooki lehelt on enamjaolt võimalik leida infot nädalavahetustel toimuvate pidude kohta. Leht reklaamib väljavalitud üritusi ning postitust *like*-des on võimalik inimestel endale selle ürituse piletid võita. Selline lähenemine on toonud neile Facebookis teatava tuntuse. Täna Tallinna Facebooki lehte jälgib üle 18 000 inimese. [12]

Puudujäägiks antud teenuse juures on ürituste piiratus, kuna enamasti reklaamitakse oma Facebooki lehel ainult pidusid ja ka need on vaid valitud. See on autori arvates tekitanud olukorra, kus Täna Tallinnat ei kasutata ürituste leidmiseks vaid pigem tasuta piletite võitmiseks.

1.2.3 Kuhuminna.ee

Kuhuminna.ee näol on tegemist veebilehega, kust on võimalik leida toimuvaid üritusi paljudes erinevates Eesti linnades: Tallinn, Tartu, Pärnu, Viljandi, Haapsalu, Paide, Jõhvi ja Võru. Samuti on nende lehel esindatud mitmete erinevate kategooriate üritused: kino, klubi/pubi/pidu, kontsert/festival, näitused/messid, teater, lastele/perele, sport/ tervis. turism, hariv/kasulik, varia, söök/jook ja tv/raadio. [13]

Puudujäägiks veebilehe juures toob autor selle, et puudub kindel info, mida edasi antakse. Näiteks kino sektsioonis kuvatakse kasutajale esmalt kõikide linnade kõikides kinodes jooksvad filmid (vt Joonis 1). Sellega seoses jõuab autor UI/UX probleemini. Kuigi veebilehel kuvatakse palju infot nii filmide, etenduste kui ka ürituste kohta, siis on info kättesaadavus kasutajale väga keeruliseks tehtud. Veebileht ei ole ka *responsive*, mis tähendab, et telefonis kasutamine on raskendatud. [14]



Joonis 1. Kuhuminna.ee kino kategooria vaade. [14]

Autori arvates ei saa Kuhuminna.ee oma üritusi Facebooki lehelt, mis tekitab olukorra, et veebileht ei ole alati ajaga kooskõlas ning väiksemad üritused jäävad lehel kuvamata. Lisaks on probleemiks ürituste hallatus, sest ühte üritust kuvatakse vahepeal nimekirjades mitmekordselt. (vt Joonis 1)

1.2.4 Minek.ee

Minek.ee on veebilehekülg, millel kuvatakse Facebookis olevaid üritusi. Valida saab nelja erineva linna vahel ning täpsustavaks filtriks saab lisada veel ka ajalise faktori. Samuti on lehele võimalus lisada ka enda Facebooki üritus ning tegemist on *responsive* veebilehega. [15]

Autorile tundub, et teenus kasutab samasugust loogikat nagu kasutatakse Kift platvormil. Facebookile antakse list erinevatest Facebooki lehtedest, kes korraldavad üritusi ning Facebook tagastab nimekirja kõikidest üritustest, mida need lehed korraldavad.

Kuigi on loodud võimalus vaadata üritusi erinevates linnades, siis antud lehe probleemiks on ürituste filtreerimatus. Veebilehel kuvatakse kõik üritused ühe listina ning ürituste varieeruvus on seinast seina. Samuti ei saa kasutaja täpsustada, mis talle huvi pakub, mis muudab sobiva ürituse leidmise kõikidest valikutest keeruliseks. [15]

1.2.5 Huvi.tallinn.ee

Huvi.tallinn.ee on veebilehekülg, kust on võimalik leida infot Tallinnas toimuvate ürituste ning huvitegevuste kohta. Leht on täielikult *responsive*, mis tagab mugava kasutuse telefonis. Autorile jäi mulje, et lehel kuvatavad üritused ei ole otseselt seotud Facebookis olevate üritustega. [16]

Üritused on jagatud erinevatesse kategooriatesse: muusika, etendused, näitus/muuseum, mess/laat, sport, kino, ööklubi/pidu, lapsele, varia. Lehel on võimalik ürituste filtreerimisel täpsustada aega, ürituste tüüpi ning linnaosa, kus üritus toimub. Samuti on lisatud otsingu võimalus ning kasutaja saab valida tasuta ja tasuliste ürituste vahel.

19.05.17 | Reede

09:00	Bastor Studio maalinäitus „LUMMUS“	Näitus, muuseum	Õpetajate Maja Galerii, Raekoja plats…
10:00	Näitus “Mitmenäoline Venemaa”	Näitus, muuseum	Tallinna vene muuseum, Pikk 29a
11:00	HANEL OLI AUTO	Varia, Etendused	NUKU TEATER, Lai 1
11:00	7. Tallinna rakenduskunsti triennaali peanäitus "Ajavahe. Time Difference"	Näitus, muuseum, La…	Eesti Tarbekunsti- ja Disainimuuseum..
14:00	Näitus VAAT(L)US	Näitus, muuseum	Vene Teater, Vabaduse Väljak 5
17:00	Miss Kevad 2017	Etendused, Varia	Tornide väljak, Rannamäe tee 3
17:00	rahvusvahelise Tallinna lillefestivali avamine 2017	Muusika, Näitus, mu…	Tornide väljaku park, Nunne tänav
17:00	rahvusvahelise Tallinna lillefestivali avamine 2017	Näitus, muuseum, Va…	Tornide väljak, Viru vljak, Kesklinna, …
17:30	Ekskursioon teatrimajas / Guided Tour	Etendused, Muusika,…	Rahvusooper Estonia, Estonia pst 4
17:30	Lõbus OGOGO legokonstreerimise meistriklass lastele vanuses 10+	Lapsele	Jõe 3, 10151 Tallinn, Эстония
18:00	MÜLLER PEAB LAHKUMA! UUSLAVASTUS!	Etendused	Rahvusraamatukogu teatrisaal, Tõnis…
19:00	Krabat	Etendused	Tallinna Linnateater, Lai 23
19:00	Lendav hollandlane (R.Wagneri ooper)	Etendused, Muusika,…	Rahvusooper Estonia, Estonia pst 4
19:00	Viimane korrus / Poslednij etaz / Последний этаж	Varia, Etendused	Vene teater, Vabaduse väljak 5

Joonis 2. Huvi.tallinn.ee ürituste vaade. [16]

Suurimaks puudujäägiks toob autor välja liigse info üleküllus ning selle kuvamise kasutajale. Kuigi antud tabelis on kogu vajalik info ürituste kohta olemas, siis on autori arvates väga keeruline sellisest jaotusest leida endale atraktiivne üritus. Kui kasutaja tuleb lehele ilma kindla teadmiseta, mida tahetakse teha, siis on keeruline üritusi avastada. (vt Joonis 2)

2. Teenuse loomine

Antud peatükis tutvustab autor Androidi platovormi ning toob välja Androidi rakendustes tihti kohatava *mystery-meat* navigeerimise probleemi. Lisaks kirjeldab autor, kuidas jõuab Kift platvormis üritus Facebookist kuni kasutaja telefonis olevasse rakendusse. Lõpetuseks toob autor välja kahele bakalaureusetöös valmivale rakendusele seatud nõuded ja analüüsib nende valmistamiseks vaja minevaid tehnoloogiaid.

2.1 Android

Androidi OS võtab nutitelefonide turul enda alla umbes 80% kogu turust, järgneb iOS umbes 18%-ga. See näitab, et Android OS-ga seadmed on jätkuvalt enimmüüdud nutitelefonid maailmas. See aga tähendab omakorda, et ringluses on väga palju erinevate OS versioonidega seadmeid. [17]

Igal aastal toimub uue Android versiooni *release*, mis toob kaasa uued tehnoloogiad ja võimalused, kuid samas peavad arendajad arvestama ka sellega, et varajasemad versioonid ei toeta kõiki uusi funktsionaalsusi. Androidi rakendust tehes peab arendaja valima minimaalse API versiooni, mida rakendus toetab. Praegusel hetkel on kõige uuemaks Androidi versiooniks 7.1, API 25, kuid kõige uuem suurem uuendus tuli Androidi seadmetele API versioon 21-ga. Sellest uuendusest toob autor näitena välja järgmised funktsionaalsused: [18, 19]

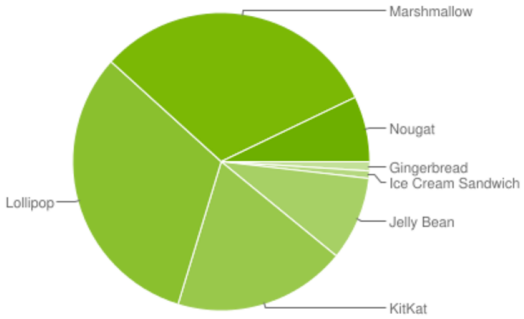
- Lihtne varjude lisamise võimekus vaadetele Z-telje abil. [20]
- *Activity transitions* ehk animatsioonid *Activity*-te vahel liikumiseks. [20]
- OpenGL - võimaldab kuvada keerukamaid animatsioone, näiteks *Ripple* effekt. [20]

Kui arendaja soovib kasutada API 21-ga tulnud lisafunktsionaalsusi on kaks valikut. Esiteks on võimalus määrata minimaalseks API versiooniks 21, mis tähendaks, et juba on 25% Androidi kasutajaid, kes ei saa rakendust kasutada. Teiseks valikuks on valida madalam minimaalne API versioon, et võimaldada suuremale hulgale kasutajatele rakenduse kättesaadavus. Sellisel juhul on vaja igasse kohta koodis, kus soovitakse uut

funktsionaalsust kasutada, lisada kontroll, mis selgitab välja, mis versioon antud seadmel on ning jooksutab siis vastavat koodi. [18] (vt Joonis 3)

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%

Data collected during a 7-day period ending on May 2, 2017.
Any versions with less than 0.1% distribution are not shown.



Joonis 3. Androidi versioonide kasutatavuse jaotus. [19]

Et tagada võimalikult suur kasutajate arv, valis autor minimaalseks versiooniks 16, mis tähendab, et rakendused on kättesaadavad umbes 98% Androidi kasutajatest. (vt Joonis 3)

2.2 *Mystery-meat* navigatsioon

'*Mystery-meat* navigatsioon' on väljend, mida esimesena kasutus 1998. aastal Vincent Flanders oma veebilehel 'Web Pages That Suck'. See viitab nuppudele või navigeerimisele, mis ei selgita ennast. Selle asemel peab kasutaja esmalt ise läbi proovima, et teada saada, kuidas see töötab. [21]

Mistery-meat navigatsioon on eriti suureks probleemiks Androidi platvormil, kus disainis on tähtsamal kohal rakenduse ilu, kuid selle tõttu kannatab kasutatavus. Heaks näiteks on alates Android 5.0 tutvustatud navigeerimisriba. Tegemist on küll väga ilusa ja puhta lahendusega, kuid tekitab kasutajale küsimusi ja ei ole üheselt mõistetav.

Kasutaja võib aru saada, et kolmnurk tähistab tagasi minekut ning võib-olla mõistab ka ringi tähendust *home* ekraanile naasmiseks, kuid ruut on sinna lisatud vaid seetõttu, et see näeb hea välja ning kasutaja saab selle funktsionaalsusest teada alles pärast sinna vajutamist. [21] (vt Joonis 4)

Navigation bar, Android Lollipop and up

Joonis 4. Android 5.0 tutvustas uut navigeerimisriba.

Samuti saab m*ystery-meat* navigatsiooni probleemiks nimetada ka Androidis väga populaarset *hamburger menu*, kus navigeerimispaneel on esmalt üldse ekraanilt väljas. Selle olukorra parandamiseks avalikustas Google 2016. aastal alternatiivi- *bottombar* navigatsioon. [21] (vt Joonis 5)

Bottom navigation bar with 4 views: mystery meat

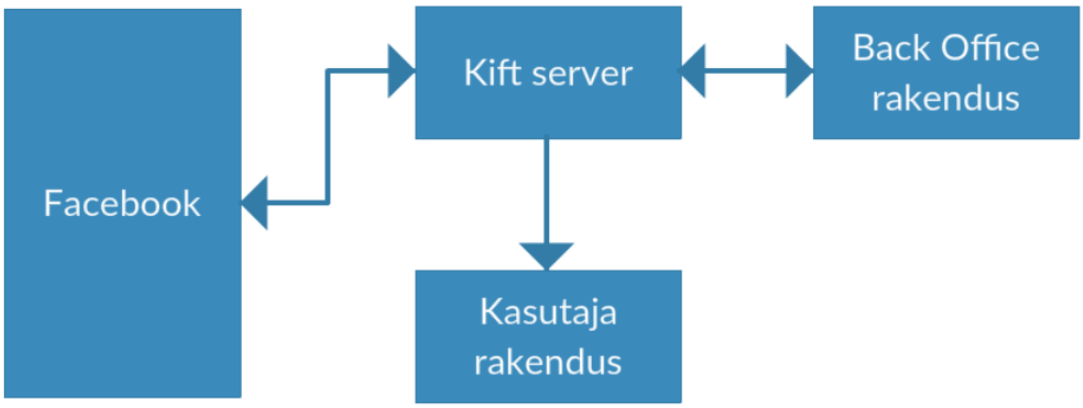
Joonis 5. 2016 avalikustas Google uue navigeerimisvõimaluse- *bottombar* navigatsioon.

Sel korral on olukord paranenud, sest navigeerimispaneel on koheselt nähtav, kuid jätkuvalt jääb kasutajale mõistatuseks, mida teised ikoonid tähistavad. Et kasutajas võimalikult vähe küsimusi tekitada on soovituslik linkidele ja nuppudele alati lisada selgitav tekst või kasutada üheselt ja selgelt mõistetavaid ikoone. [17]

2.3 Kift platvorm

Kift platvorm on täiesti eraldiseisev üksus, mis kasutab Facebook Graph API-d, et saada informatsiooni toimuvate ürituste jaoks. See koosneb kolmest osast. [44] (vt Joonis 6)

- Kift server
- *Back office* rakendus
- Kasutaja rakendus



Joonis 6. Kift platvormi arhitektuur.

Järgnevalt kirjeldab autor kuidas jõuab üritus Facebooki lehelt Kift rakendusse. (vt Joonis 6)

- Kift serveris on valmis kirjutatud skriptid, mis iga kindla aja tagant saadavad Facebook Graph API-le päringu, milles on kaasa antud nimekiri Facebooki lehtede id-dega, mis korraldavad üritusi.
- Facebook tagastab Kift serverile nimekirja kõikidest soovitud üritustest.
- Need üritused, mis saadakse Facebooki lehtedelt, millele vaikimisi kategooriat lisatud ei ole, jäävad kasutajale esialgu kättesaamatuks ning lähevad lähevad ootelisti, mida hallatakse *back office* rakenduse kaudu.
- Üritused, mis saadakse Facebooki lehtedelt ning millele on määratud vaikimisi kategooria, jagatakse automaatselt ära ning kuvatakse koheselt *client* rakenduses.

- *Back office* rakenduses lisatakse kõikidele ootelistis olevatele üritustele kategooriad ning seega tehakse nad kasutajatele nähtavaks.

2.4 Rakendustele esitatud nõuded

Järgnevas kahes alapeatükis nimetab autor mõlemale rakendusele esitatud nõuded.

2.4.1 *Back office* rakendusele esitatud nõuded

Back office rakenduse määramisel tugines autor arusaamale, et süsteemi kasutamine peab olema võimalikult kiire ja mugav.

- Et säilitada turvalisust tuleb rakenduse kasutamiseks sisse logida, kas kasutajanime ja parooliga või näpujälje autoriseerimisega.
- Rakenduse avamisel kuvatakse kasutajale kõik kategoriseerimata üritused.
- Kasutajal on võimalus vaadata üritusi, mis on juba kindlatesse kategooriatesse määratud.
- Kasutaja saab määrata üritusele ühe või rohkem kategooriaid.
- Kasutaja saab muuta oma valikut pärast ürituse salvestamist.
- Kasutaja saab ürituse *dismiss*-da, et seda ei kuvataks üheski kategoorias.
- Kasutajal on võimalus näha kõiki *dismiss*-tud üritusi ning saab neid uuesti kategooriatesse lisada.
- Ürituste listis kuvatakse ürituse kohta täpsustavat informatsiooni:
    - ürituse nimi
    - ürituse asukoht
    - nupp ürituse Facebooki lehele minemiseks

- Üritusele vajutades avaneb vaade, kus kasutaja näeb ürituse kohta rohkem informatsiooni:
    - ürituse nimi
    - ürituse asukoht
    - ürituse toimumise aeg
    - ürituse kirjeldus
    - nupp ürituse Facebooki lehele minemiseks
- Kasutajal on võimekus muuta ürituse kohta käivat infot:
    - ürituse nimi
    - ürituse asukoht
    - ürituse kirjeldus
- Kasutaja näeb infot kõikide üritusi korraldavate Facebooki lehtede kohta, mis on Kift serverisse lisatud.
- Kasutaja saab lisada Kift serverisse Facebooki lehtede id-sid.
- Kasutaja saab määrata Facebooki lehe vaikimisi kategooria, mis tähendab, et kõik üritused, mis sealt lehelt tulevad, saavad automaatselt endale selle kategooria külge.
- Kasutaja saab muuta juba lisatud Facebooki lehtede infot.
- Kasutaja saab kustutada olemasolevaid Facebooki lehti Kift serverist.
- Kasutaja saab käivitada Kift serveris oleva skripti, mis alustab Facebookist ürituste *fetch*-imist**.**

2.4.2 *Client* rakendusele esitatud nõuded

Client rakenduse loomisel arvestab autor juba olemasolevate teenuste positiivsete ja negatiivsete külgedega.

- Rakendus peab vältima my*stery-meat* navigeerimise probleemi. Kasutajal ei tohi hetkekski tekkida küsimust, kuidas rakenduses navigeerida.
- Rakenduse kasutamiseks ei pea kasutaja endale kontot looma.
- Rakendus on võimeline vastu võtma *notification*-eid ja neid kasutajale kuvama.
- Kasutajal on võimalik filtreerida üritusi 4 kategooria vahel:
    - *Dance* kategooria, mille alla kuuluvad enamasti peod. Näiteks üritused klubides.
    - *Concert* kategooria, mille alla kuuluvad kontserdid. Alates väikestest muusikakooli inimeste kokkutulekutest kuni suurte kontsertideni.
    - *Sport* kategooria, mille alla kuuluvad kõik spordiga seotud üritused. Näiteks jalgpallimängud ning rahvajooksud.
    - *Kift* kategooria alla kuuluvad kõikide ülalnimetatute kategooriate kõige erilisemad üritused, mis on välja valitud platvormi haldajate poolt. Samuti võib siit kategooriast leida teisi huvitavaid üritusi. Näiteks toob autor filmiõhtud ja pannkoogihommikud.
- Rakenduse avamisel kuvatakse esmalt kasutajale kõik üritused, mis on Kift kategoorias.
- Listis oleval ürituse elemendil on kuvatud järgnev info:
    - ürituse nimi
    - ürituse asukoht
    - ürituse toimumise aeg

        - ürituse kategooriad
        - Kui Facebooki üritusel on üle 100 inimese pannud *interested*, siis kuvatakse lisainfo, et inimesed on sellest üritusest huvitatud.
        - Kui üritus algab homme, täna või on praegu käimas, siis kuvatakse selle kohta täpsustav teade.
- Kasutaja saab minna kaardi vaatesse, kus kuvatakse kõik üritused.
- Kasutaja saab kaardil leida oma asukoha.
- Kaardil ürituse peal vajutades peab avanema täpsustav info ürituse kohta, mis sisaldab endas järgnevat infot:
    - ürituse nimi
    - ürituse asukoht
    - ürituse kategooriad
    - Kui üritus algab homme, täna või on praegu käimas, siis kuvatakse selle kohta täpsustav info.
- Kaardil oleva täpsustava info peal vajutades peab avanema ürituse vaade, kus on kogu info ürituse kohta.
- Listis oleva ürituse peal vajutades peab avanema ürituse vaade, kus on kogu info ürituse kohta.
- Info, mis peab olema kuvatud ürituse vaates:
    - Ürituse nimi
    - Ürituse asukoht
    - Ürituse toimumise aeg

- Kui üritusel on üle 10 inimese *interested* või *going*, siis kuvama vastavalt ürituse *interested* ja *going* arvud.
- Kui üritus on toimumas homme, täna või on praegu käimas, siis näidatakse selle kohta täpsustavat teadet:
    - Ürituse kirjeldus
    - Ürituse kategooriad
    - Nupp, mis suunab kasutaja kaardile, kus on fokusseeritud antud üritus.
    - Nupp, mis suunab kasutaja ürituse Facebooki lehele.
    - Nupp, millega saab üritust jagada oma sõpradele.

2.5 Valitud tehnoloogiate tutvustus ja analüüs

Järgnevas peatükis räägib autor kasutatud tehnoloogiatest. Kõigepealt tutvustab autor rakenduste valmimiseks kasutatud programmeerimiskeelt ning selgitab valitud arhitektuurilist lahendust. Seejärel analüüsib autor täpsemalt spetsiifilisemate tehnoloogiate valikuid.

2.5.1 Kotlin

Kotlin on programmeerimiskeel, mis avalikustati 2011. aastal ning on loodud tehnoloogiaettevõtte JetBrains poolt. Kotlin on otsene konkurent Javale ning põhjuseid, miks uus keel tehti on mitmeid. Esiteks, Java sai alguse üle 20. aasta tagasi ning Java põhimõtted on, et iga uuem versioon peab toetama vana versiooni. See on kaasa toonud olukorra, kus väga palju uut funktsionaalsust jääb lisamata, sest seda ei ole võimalik panna koos töötama vanemate versioonidega. Lisaks toob autor välja, et Java sisaldab väga palju *boiler-plate* koodi, mis Kotlinis on miinimumini viidud. [23]

Kuigi Kotlin avalikustati juba 2011. aastal, siis tegelikkuses jõudis rahvani stabiilne versioon nimega Kotlin 1.0 alles 2016. aastal. Kotlin toob keelena endaga kaasa palju võimalusi: [22]

- *Null safety*
- Laiendavad funktsioonid
- Lambdade kasutamine
- Data klassid
- Muutuja tüüpi ei ole vaja määrata, Kotlin saab muutujale väärtuse andmisest aru, mis tüüpi elemendiga on tegemist.
- Vähemalt 20% vähem koodi võrreldes Java koodiga, mis on sama funktsionaalsusega.
- Funktsioonid, mille parameetriteks saab lisada funktsioone ja mis tagastavad funktsioone.

Esiteks toob autor välja n*ull safety,* mis tähendab, et algselt on kõik muutajad mitte *null*-d ning kompileerija ei lase arendajal luua mitte lähestatud muutujat. Kui kasutaja siiski soovib luua lähestamata muutujat peab ta selle iseseisvalt deklareerima. Javas tuleb muutuja kasutamisel selle ette enamasti kirjutada *null check*, mis kontrollib, kas muutuja on ikka lähestatud. [22] (vt Joonis 7)

```
if (nullable != null) nullable.doStuff()
```

Joonis 7. Javas tuleb enne objekti kasutamist kontrollida, kas objekt ei ole *null.*

Kotlin kasutab sellistes olukordades *safe call operator*-t, mis annab arendajale võimaluse lihtsalt objektile lisada küsimärgi, mis tähistab protseduuri, et objekti funktsioon läheb täitmisele ainult siis, kui objekt ei ole *null* väärtusega. [22] (vt Joonis 8)

```
nullable?.doStuff()
```

Joonis 8. Kotlinis saab objekti lähestatust kontrollida küsimärgiga.

Teiseks on väga suur roll koodi mahul. Kotlinis on näiteks kasutusel Data klassid ja lambdad, mis tähendab, et väga palju *boiler-plate* koodi jääb kirjutamata. [23] (vt Joonis 9, 10)

```
townNameTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onTownClick(pagerPos);
    }
});
```

Joonis 9. Javas elemendile *onClick listeneri* lisamine.

Kotlini Data klassid võimaldavad kasutajal jätta kirjutamata kõik *getterid* ja *setterid* Klassile, Kotlin saab sellega ise hakkama. [22]

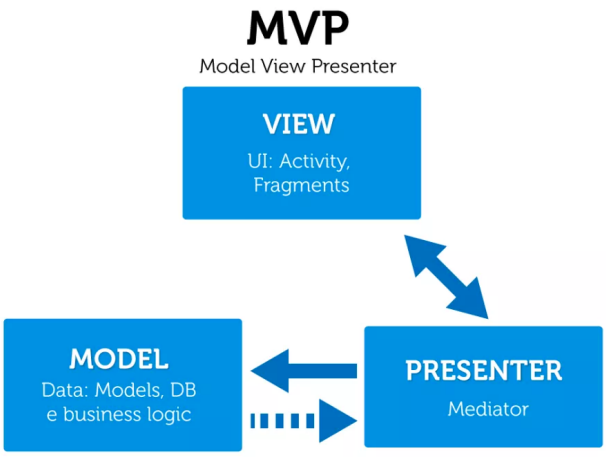
```
info_layout.setOnClickListener {
    operator!!.onMoreInfoClick(event)
}
```

Joonis 10. Kotlinis elemendile *onClick listeneri* lisamine.

Kotlinil on võrreldes Javaga palju eeliseid, kuid kuna tegemist on siiski võrdlemiselt noore programmeerimiskeelega, siis võib jätkuvalt tekkida olukordi, mida Kotlin ei ole võimeline lahendama. Selliste olukordade vältimiseks on Kotlin üles ehitatud selliselt, et kui mõnda probleemi ei saa lahendada, saab selle kirjutada ka Javas, mis tähendab, et Kotlin ja Java on omavahel täielikult koostöötavad. [23]

2.5.2 MVP arhitektuur

Autor valis mõlema rakenduse arhitektuuriliseks lahenduseks MVP, mis tähendab Model-View-Presenter. MVP lahenduses on 3 elementi *view* ehk vaade, *presenter* ehk vahemees ja *model* ehk info hoiustaja. [25]



Joonis 11. Model-View-Presenter arhitektuur.

View, mis tavaliselt väljendub Androidis *Activity* näol, ainukeseks ülesandeks on kuvada kasutajale infot ning anda *presenterile* teada, kui kasutaja midagi ekraanil teeb. [25] (vt Joonis 11)

Teiseks elemendiks on *presenter*, mille põhiülesandeks on olla vahemeheks *view* ja *modeli* vahel. *presenter* saab info *model*-lt ning ütleb *view*-le kuidas seda kuvada. Samuti on *presenter* ülesandeks otsustada, mis juhtub kui kasutaja teeb midagi *view*-s. [25] (vt Joonis 11)

Viimaseks elemendiks on *model*. Hea loogikaga rakenduses on *modeli* ainsaks ülesandeks hoiustada infot ning alustada ja hallata API päringuid. [25] (vt Joonis 11)

Autor valis MVP, sest selline arhitektuuriline lahendus toob kaasa *separation of concerns*-i, mis tähendab, et rakenduse erinevatel osadel on oma kindel ülesanne. Miski ei pea tegema mitut asja. Selline lahendus muudab koodi loetavust lihtsamaks, võimaldab projekti skaleeruvuse, minimaliseerib *memory leaks*-d, ning üleüldiselt parandab koodi hooldatavust. [26]

2.5.3 RxJava 2

RxJava on Java realisatsioon asünkroonsest programmeerimisest, millega saab luua jälgitavaid *sequence*. RxJava 2 võimaldab asendada kõik *callback* klassid lihtsa *observable* ja *subscribe* lahendusega. Samuti ei ole vaja mõelda *AsyncTaskide* loomisele, see kõik toimub iseseisvalt *observable*-ga. [27, 28]

```
fun getEvents(listener: DataListener): Disposable? {
    val dataSource: Observable<ResponseBody> = eventService!!.getData()
    return dataSource.subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe({
                responseBody ->
                val responseType = object : TypeToken<BaseResponse>() {}.type
                val baseResponse: BaseResponse = gson.fromJson(responseBody.string(), responseType)
                listener.onSuccess(baseResponse.response!!)
            }, { error ->
                listener.onError(error.message!!)
            })
}
```

Joonis 12. RxJava 2-s API päringu tegemine.

RxJava 2-s toimub asünkroonse päringu tegemine järgnevalt: (vt Joonis 12)

- Kasutaja loob *Observable* objekti.
- Kasutaja määrab, millises *threadis Observable* välja kutsutakse. (vt Joonis 12 *subscribeOn call*)
- Kasutaja määrab, millises *threadis Observable* streami jälgib. (vt Joonis 12 *observeOn* call)
- Kasutaja määrab, mis juhtub, kui päring on lõppenud. (vt Joonis 12 subscribe *call*)

Autor integreeris oma rakendustes RxJava 2 Retrofit-ga, et muuta API päringute tegemine lihtsaks ja arusaadavaks.

2.5.4 Retrofit ja Gson

Retrofiti ja Gsoni kasutatakse Androidi rakenduste arendamisel, et lihtsustada API päringute tegemist.

Retrofiti ülesandeks on muuta HTTP API Java *interfaceks*. Tegemist on REST kliendiga, mis lihtsustab JSON-i kätte saamist päringust ning JSON objekti saatmist serverisse. [29]

Gson on Java *library*, mida kasutatakse, et konverteerida Java objektid JSON-ks ja vastupidi. See lihtsustab märkimisväärselt API päringute tulemusel saadud info kasutamist programmis. Samuti ei pea Gsoni kasutades muretsema API päringuga kaasa saadetava JSONi pärast. Arendaja annab objekti kaasa ning Retrofiti ja Gsoni kooslusel muudetakse see vastavaks JSON objektiks. [30]

```
fun <T> createRetroiftService(klass: Class<T>, client: OkHttpClient, baseUrl: String, gson: Gson): T {
    val retrofit = Retrofit.Builder()
            .client(client)
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create(gson))
            .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
            .build()
    return retrofit.create(klass)
}
```

Joonis 13. Retrofiti teenuse loomine.

2.5.5 *Firebase Cloud Messaging*

Firebase Cloud Messaging ehk FCM on *cross-platform* teenus, mis võimaldab saata sõnumeid kasutajate nutiseadmetesse. [30]

Joonis 14. Firebase Cloud Message implementatsioon Kift platvormis.

Järgnevalt kirjeldab autor, kuidas jõuab sõnum Kift serverist kasutaja telefonis olevasse rakendusse. (vt Joonis 14)

- Esmalt saadetakse Kift serverist POST päring FCM-i. Selles päringus on ära täpsustatud rakenduse *Authorization key*, mille järgi FCM saab aru, millisele rakendusele sõnumit edasi saata. Samuti on päringu *body*-s informatsioon, mida soovitakse kasutaja telefonis kuvada. [31] (vt Joonis 15)

```
{
  "to": "/topics/firebase_topic",
  "data": {
    "title" : "Kift",
    "body": "It's almost Friday, find the most Kift events"
   }
}
```

Joonis 15. FCM teenusele saadetava POST päringu *body*.

- FCM töötleb päringu:
    - väärtus ‘to’ näitab millisele kasutajagrupile on sõnum mõeldud. Kift-i näitel registreeritakse kõik kasutajad ‘firebase_topic’ kategooriasse. [31]
    - objekt ‘data’ sees on kogu info, mis peab jõudma kasutaja nutiseadmesse. Sinna saab juurde lisada veel peale pealkirja ja kirjelduse ka oma *custom* informatsiooni. [31]
- FCM saadab päringu kasutaja rakendusse. [31]

- Rakenduses on deklareeritud *service*-na FirebaseMessagingService, mis taustal kogu aeg jälgib, kas antud rakendusse on saadetud sõnumeid. [31]
- Kui sõnum jõuab kasutaja rakendusse, siis esmalt on tegemist *RemoteMessage* objektiga. [31]
- Sellest objektist on vaja välja *parse*-da soovitud info ning seejärel ette valmistada sõnumi objekt, mida kasutajale kuvatakse. (vt Joonis 16)
- Kui sõnumi objekt on valmis, saab *NotificationManager.notify call*-ga kuvada kasutajale sõnumit. (vt joonis 16)

```
private fun buildNotification(message: RemoteMessage) {
    val notification: Notification?
    val intent = Intent(this, MainActivity::class.java)
    intent.putExtra(ARG_COMING_FROM_NOTIFICATION, true)
    val pendingIntent = PendingIntent.getActivity(this, 0 /* request code */, intent, Pe

    if (Build.VERSION.SDK_INT >= 21) {
        notification = makeNotificationSDK21orHigher(applicationContext, pendingIntent,
                message.data[NOTIFICATION_TITLE]!!, message.data[NOTIFICATION_BODY]!!)
    } else {
        notification = makeNotificationSDK16orHigher(applicationContext, pendingIntent,
                message.data[NOTIFICATION_TITLE]!!, message.data[NOTIFICATION_BODY]!!)
    }

    val notificationManager = this.applicationContext.getSystemService(Context.NOTIFICAT
    notificationManager.notify(0, notification)

}
```

Joonis 16. FCM-st jõuab rakendusse RemoteMessage, mida on vaja töödelda, et see oleks kasutajale kuvatav.

3. Valminud rakendused

Bakalaureusetöö käigus valmis kaks rakendust. Kift *back office* rakendus on Kift projektiga seotud inimestele välja jagatud ning toimib praegu eraldiseisvas test keskkonnas. Kift *client* rakendus on antud hetkel Google Play Store Alpha testimises ja ligipääs on antud kõigile tuttavatele ning sõpradele, et saada esmast tagasisidet. Kift *client* rakenduse loomisel võeti arvesse teiste ürituste leidmise teenust pakkuvate lehtede tugevaid ja nõrku külgi ning loodi rakendus, kust on võimalik lihtsalt ja arusaadavalt kätte saada kogu vajalik info toimuvate ürituste kohta. Järgnevas peatükis toob autor välja kasutatud töövahendi, nimetab *library*-d mida projektides kasutati ja võrdleb valminud Kift *client* rakendust olemasolevate lahendustega. Lõpetuseks tutvustab autor edasiarendamise suundasid.

3.1 Kasutatud töövahend ja *library*-d

Autor kasutas rakenduste arendamiseks Android Studio-t. API päringute testimiseks kasutati programmi nimega Postman. Järgnevalt toob autor välja tähtsamad *library*-d, mida mõlemas projektis kasutati: [32]

- RxJava2.rxkotlin - asünkroonsete päringute tegemiseks Kotlinis [34]
- RxJava2.rxandroid - asünkroonsete päringute tegemiseks Androidis [34]
- Retrofit2 & Retrofit2.gson- API päringute lihtsustamiseks [29]
- Chrisjenx:calligraphy - *custom* fontide kasutamiseks [35]
- Picasso - piltide kuvamine vaadetes, omades piltide URL-i [36]
- Kasparpeterson.simplemvp - MVP struktuuri põhi [24]
- Firebase-messaging - FCM, *push notificationite* vastu võtmiseks [31]
- Play-services-maps - Google mapsi kasutamiseks [39]
- Lottie - Laadimisanimatsioonide jaoks [37]

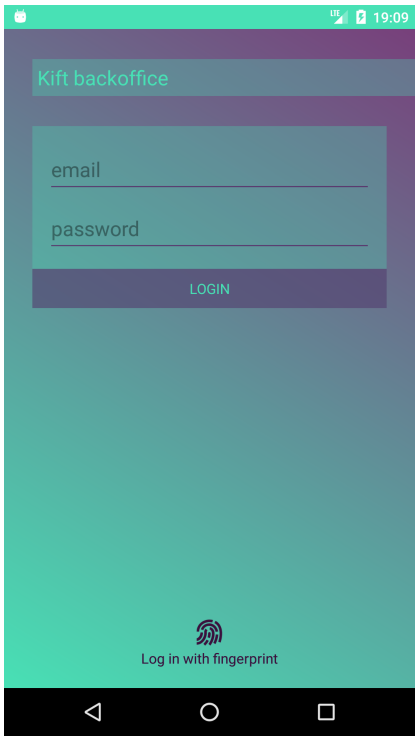
- Grantland:autofittextview - Võimaldab teksti alati ekraanile ära mahutada [38]

3.2 Kift *back office* rakenduse põhilised funktsionaalsused

Järgnevas alapeaktükis toob autor välja Kift *back office* rakenduse peamised funktsionaalsused ja *user flow*-d. Rakenduse kood on kättesaadav https://bitbucket.org/eerikk/kift-backoffice.

3.2.1 Sisselogimine

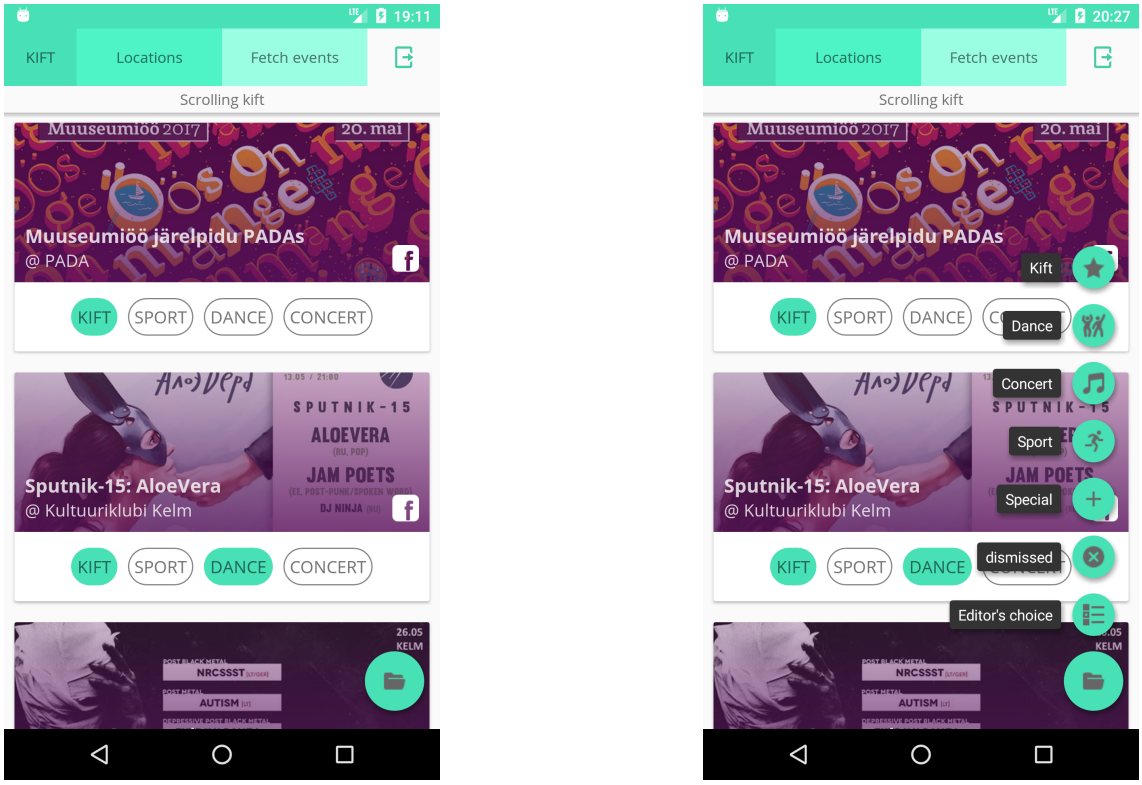
Sisse logimiseks on kasutajal kaks võimalust. Esiteks traditsiooniline kasutajanime ja parooliga sisse logimine ning teiseks on lisatud ka *fingerprint authentication*, mis muudab sisse logimise kiiremaks ja mugavamaks. Rakenduse avamisel kontrollitakse automaatselt, kas telefonil on näpujälje tuvastuse võimekus ning, kas kasutaja on mõne näpujälje autoriseerinud. Kui eelnevalt nimetatud tingimused on täidetud, kuvatakse ekraani allosas kasutajale *fingerprint* logo, mis annab märku, et rakendusse saab sisse logida näpujäljega. Kui nutiseadmel ei ole sellist funktsionaalsust, siis ekraani allosa jääb tühjaks. (vt Joonis 17)



Joonis 17. Kift *back office* login vaade.

3.2.2 Ürituste kuvamine listis

Autori eesmärgiks oli luua vaade, mis on võimalikult lihtsalt hoomatav, kuid täidab kõik nõuded ja muudab ürituste haldamise kiireks ja mugavaks. Selles vaates (vt Joonis 18) on võimalik *scrollida* kindlas kategoorias olevaid üritusi. Samuti on võimalik kiiresti määrata ürituse kategooriaid ning avada ürituse Facebooki leht. Lisaks saab aktiveerida serveris oleva skripti, mis kogub Facebookist soovitud üritusi. Ürituse spetsiifilisemaks muutmiseks tuleb vajutada ürituse peale. (vt joonis 18)

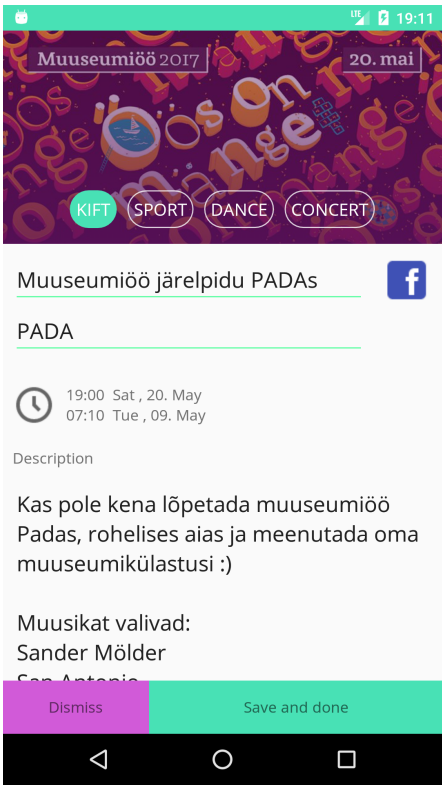


(a) (b)

Joonis 18. Kift *back office* ürituste *scrollitav* nimekiri (a) ja spetsiifilise kategooria ürituste vaatamine (b).

3.2.3 Ürituse vaade

Ürituse vaates on kasutajal võimalik sarnaselt listi vaatega kiiresti määrata ürituse kategooriaid ning näha üritust Facebookis. Lisaks saab selles vaates teha spetsiifilisemaid muudatusi. Näiteks on võimalik muuta ürituse nime, asukohta ning ka kirjeldust. (vt Joonis 19)

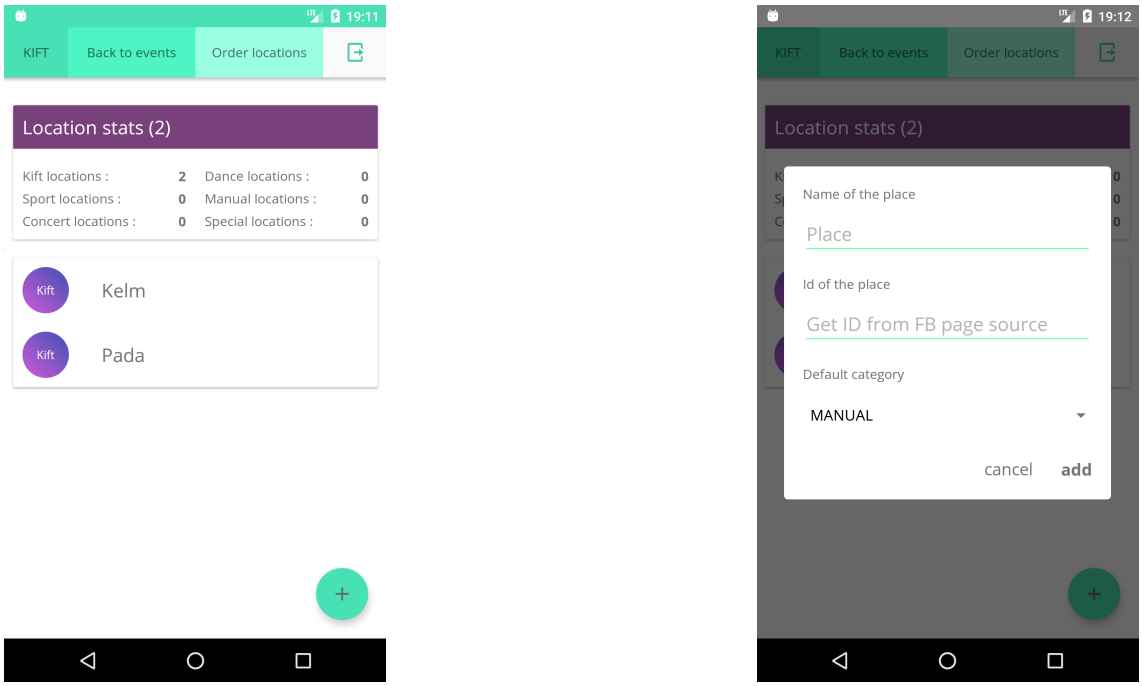


Joonis 19. Kift *back office* ürituse vaade.

3.2.4 Üritusi korraldavate Facebooki lehtede vaade

Antud vaates on võimalik näha ülevaadet kõikidest Facebooki lehtedest, kust Kift server üritusi *fetchib*. Vaate ülemises osas on toodud statistika kõikide lehtede kohta ning selle all on nimekiri kõikidest kohtadest ja nende kategooriatest. (vt Joonis 20(a))

All paremas nurgas on pluss märgiga FAB, millele vajutades avaneb uue ürituste korraldaja lisamise vaade. Antud vaates peab kasutaja sisestama Facebooki lehe nime ja id, et Kift serveris jooksev skript saaks õige info Facebook Graph Api-le saata. Samuti on võimalik määrata vaikimisi kategooria, mis läheb külge kõikidele üritustele, mis on saadud sellelt Facebooki lehelt. (vt Joonis 20(b))



(a) (b)

Joonis 20. Kift *back office* Facebooki ürituste korraldajate ülevaade (a) ning uue ürituste korraldamise lehe lisamine (b).

3.2.5 Kift *back office* rakenduse vastavus esitatud nõuetele

Järgnevas tabelis toob autor välja Kift *back office* rakendusele esitatud nõuded ja nende täidetuse. Kõik esitatud nõuded said täidetud, täpsustava informatsiooni leiab järgnevast tabelist. (vt Tabel 1)

Tabel 1. Kift *back office* nõuete täidetus.

Nõue	**Nõuetele vastavus**
Et säilitada turvalisust tuleb rakenduse kasutamiseks sisse logida kas kasutajanime ja parooliga või *fingerprint* autoriseerimisega.	Täidetud, rakenduse kasutamiseks peab kasutaja ennast autoriseerima, kui nutiseadmel on näpujälje tuvastus on kasutajal võimalik ennast identifitseerida sellega.
Rakenduse avamisel kuvatakse kasutajale kõik kategoriseerimata üritused.	Täidetud
Kasutajal on võimalus vaadata üritusi, mis on juba kindlatesse kategooriatesse määratud.	Täidetud
Kasutaja saab määrata üritusele ühe või rohkem kategooriaid.	Täidetud
Kasutaja saab muuta oma valikut pärast ürituse salvestamist.	Täidetud, ürituse kategooriat saab muuta pärast ürituse salvestamist.

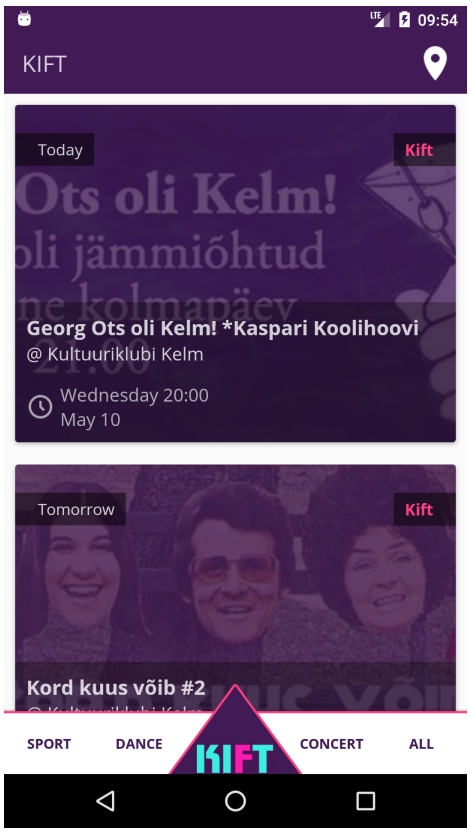
Nõue	**Nõuetele vastavus**
Kasutaja saab ürituse *dismiss*-da, et seda ei kuvataks üheski kategoorias.	Täidetud
Kasutajal on võimalus näha kõiki *dismiss*-tud üritusi ning saab neid uuesti kategooriatesse lisada.	Täidetud
Ürituste listis näeb ürituse kohta täpsustavat informatsiooni.	Täidetud, ürituste listis on iga ürituse juures lisainfona ürituse nimi, asukoht ja nupp, mis suunab ürituse Facebooki lehele.
Üritusele vajutades avaneb vaade, kus kasutaja näeb ürituse kohta rohkem informatsiooni.	Täidetud, kindla ürituse vaates on kasutajale näha ürituse nimi, asukoht, toimumise aeg, kirjeldus ning nupp, mis suunab ürituse Facebooki lehele.
Kasutajal on võimekus muuta ürituse kohta käivat infot.	Täidetud, kasutaja saab muuta ürituse nime, asukohta ning kirjeldust.
Kasutaja näeb infot kõikide üritusi korraldavate Facebooki lehtede kohta, mis on Kift serverisse lisatud.	Täidetud, üritusi korraldavate Facebooki lehtede vaates on kasutajal võimalik näha üldist statistikat kõikide lehtede kohta. Samuti on samas vaates listina kuvatud ka kõik kasutusel olevad Facebooki lehed ja nende vaikimisi kategooria.
Kasutaja saab lisada Facebooki lehti, mis korraldavad üritusi.	Täidetud
Kasutaja saab määrata Facebooki lehe vaikimisi kategooria, mis tähendab, et kõik üritused, mis sealt lehelt tulevad saavad automaatselt endale selle kategooria külge.	Täidetud, kasutaja saab jätta vaikimisi kategooria manuaalseks, mis tähendab, et kõik üritused, mis sealt tulevad lähevad ootelisti. Teiseks võimaluseks on määrata kindel kategooria, siis lähevad selle Facebooki lehe üritused automaatselt valitud kategooriasse.
Kasutaja saab muuta juba lisatud Facebooki lehtede infot.	Täidetud, kasutaja saab muuta olemasoleva Facebooki lehe nime ning vaikimisi kateoogriat.
Kasutaja saab kustutada olemasolevaid Facebooki lehti Kift serverist.	Täidetud
Kasutaja saab käivitada Kift serveris oleva skripti, mis alustab Facebookist ürituste *fetch*-imist.	Täidetud, rakenduses on võimalik alustada ürituste *fetch*-imist Facebookist, ning kui toiminguga ollakse lõpule jõutud antakse sellest Android *toast*-ga teada.

3.3 Kift *client* rakenduse põhilised funktsionaalsused

Järgnevas alapeaktükis toob autor välja Kift *client* rakenduse peamised funktsionaalsused ja *user flow*-d**.** Rakenduse kood on kättesaadav https://bitbucket.org/eerikk/kift-client.

3.3.1 Ürituste vaade listina

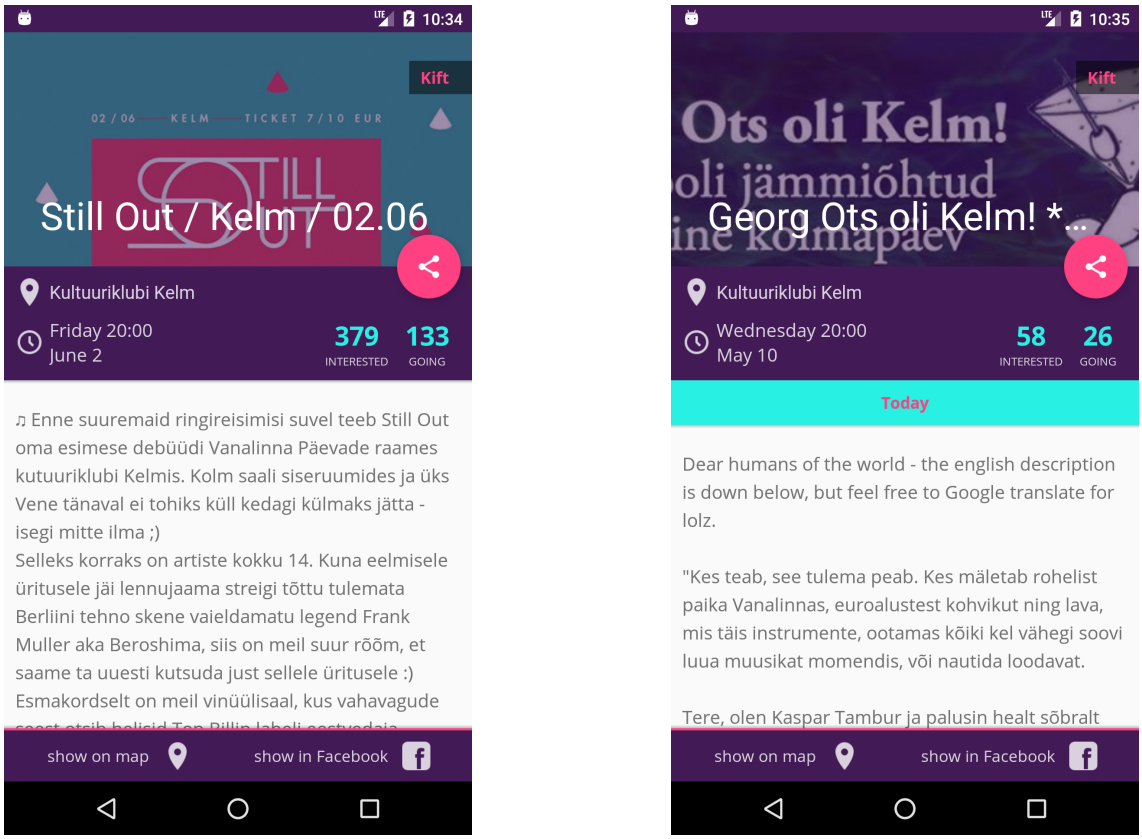
Autori eesmärk oli luua vaade, kus kasutajal on võimalus lihtsalt ja kompaktselt saada infot toimuvate ürituste kohta. Ekraanil kuvatakse nimekirja üritustest, mis on lähiajal toimumas. Samuti on kasutajal võimalik filtreerida üritusi kindla kategooria alusel ning liikuda teise vaatesse, kus saab näha üritusi kaardi peal. Vajutades listis olevale üritusele suunatakse kasutaja ürituse vaatesse. (vt Joonis 21)



Joonis 21. Kift *client* rakenduse ürituste vaade listina.

3.3.2 Ürituse vaade

Ürituse vaates on kasutajal võimalik näha täpsustavat infot ürituse kohta. Kui üritus on toimumas lähiajal kuvatakse selle kohta ka teade (vt Joonis 22(b)). Samuti on kasutajal võimalus vaadata, kus üritus toimub ning uurida täpsustavat infot ürituse Facebooki lehelt. Lisaks on võimalik üritust jagada ka oma sõpradele. (vt Joonis 22)

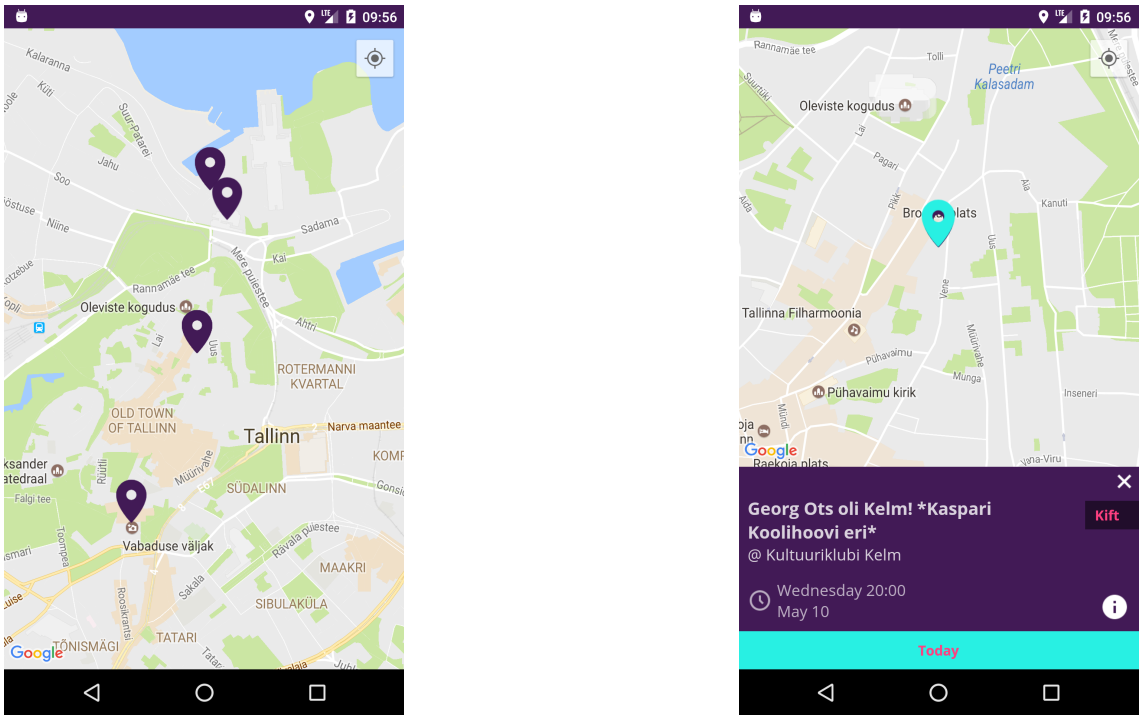


(a) (b)

Joonis 22. Kift *client* rakenduse ürituse vaade (a) ning ürituse vaade täpsustava informatsiooniga lähiajal toimuva ürituse kohta (b).

3.3.3 Ürituste vaatamine kaardil

Antud vaates on kasutajal võimalik näha toimuvaid üritusi kaardil. Üritusele vajutades avaneb lühiinfo ürituse kohta. See loob alternatiivse võimaluse ürituste leidmiseks, kui kasutajal pole kindlat mõtet, mida teha vaid soovib leida midagi, mis on talle lähedal. (vt Joonis 23)

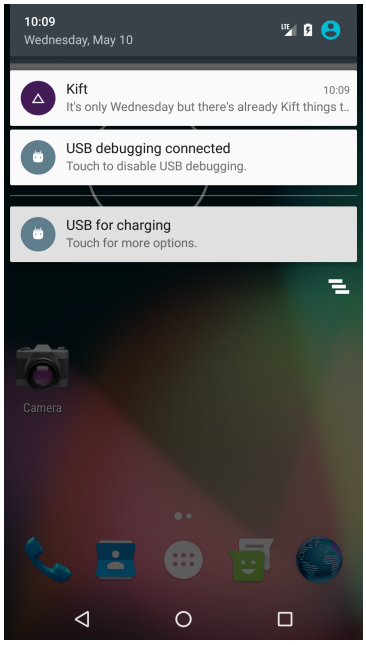


(a) (b)

Joonis 23. Kift *client* rakenduse kaardi vaade (a), ning vaade kus kaardil on fokusseeritud üritus (b).

3.3.4 *Notification*-ite vastu võtmine

Rakendus on võimeline vastu võtma ning kasutajale kuvama saadetud *notification*-d. See funktsionaalsus hoiab kasutajad rakenduse juures ning tekitab võimaluse seadmetesse erineva sisuga teateid saata. (vt Joonis 24)



Joonis 24. Rakendus on võimeline kuvama saadud *notification*-eid.

3.3.4 Kift *client* rakenduse vastavus esitatud nõuetele

Järgnevas tabelis toob autor välja Kift *client* rakendusele esitatud nõuded. Kõik esitatud nõuded said täidetud, täpsustava informatsiooni leiab järgnevast tabelist. (vt Tabel 2)

Tabel 2. Kift *client* rakendusele esitatud nõuded ja nende täidetus.

Nõue	**Nõuetele vastavus**
Rakendus peab vältima *mystery-meat* navigeerimise probleemi. Kasutajal ei tohi hetkekski tekkida küsimust, kuidas rakenduses navigeerida.	Täidetud, rakenduse valmistamisel välditi *mystery-meat* probleemi. Enamus navigeerimisnuppudest on selgitava tekstiga. Ainult kahe nuppu osas otsustas autor vaid ikooni kasuks - kaardivaatesse minemine ning ürituse jagamine.
Rakenduse kasutamiseks ei pea kasutaja endale kontot looma.	Täidetud
Rakenduse avamisel kuvatakse esmalt kasutajale kõik üritused, mis on Kift kategoorias.	Täidetud
Rakendus on võimeline vastu võtma *notification*-eid ja neid kasutajale kuvama.	Täidetud
Kasutajal on võimalik filtreerida üritusi 4 kategooria vahel.	Täidetud, rakenduses on võimalik üritusi näha 4 kategoorias ning ilma kategooriata vaates.
Listis oleval ürituse elemendil on kuvatud järgnev info.	Täidetud, listis oleval ürituse elemendil kuvatakse ürituse nimi, asukoht, kellaaeg ning kategooriad.
Kui Facebooki üritusel on üle 100 inimese pannud *interested*, siis kuvatakse info, et inimesed on sellest üritusest huvitatud.	Täidetud, info kuvatakse kasutajale ürituste listi vaates.
Kasutaja saab minna kaardi vaatesse, kus kuvatakse kõik üritused.	Täidetud
Kasutaja saab kaardil leida oma asukoha.	Täidetud, olles kaardivaates ning vajutades *default* Google Maps ‘näita mu asukohta’ nuppu, fokuseerib kaart kasutaja asukoha.
Kaardi vaates ürituse peale vajutades peab avanema täpsustav info ürituse kohta, mis sisaldab endas järgnevat infot.	Täidetud, kaardil ürituse peal vajutades näeb kasutaja ürituse nime, asukohta, kategooriaid ning kui üritus on toimumas lähiajal, siis selle kohta ka täpsustavat teadet.
Kaardil oleva täpsustava info peal vajutades peab avanema ürituse vaade, kus on kogu info ürituse kohta.	Täidetud

Nõue	Nõuetele vastavus
Listis oleva ürituse peal vajutades peab avanema ürituse vaade, kus on kogu info ürituse kohta.	Täidetud
Info, mis peab olema kuvatud ürituse vaates.	Täidetud, ürituse vaates näeb kasutaja: • Ürituse nimi • Ürituse askoht • Ürituse toimumise aeg • Kui üritusel on üle 10 inimese *interested* või *going*, siis kuvatakse vastavalt ürituse *interested* ja *going* arvud • Kui üritus on toimumas homme, täna või on praegu käimas, siis täpsustav info selle kohta • Ürituse kirjeldus • Ürituse kategooriad • Nupp, mis suunab kasutaja kaardile, kus on fokusseeritud antud üritus. • Nupp, mis suunab kasutaja ürituse Facebooki lehele • Nupp, millega saab üritust jagada oma sõpradele

3.4 Kift *client* rakendus võrreldes olemasolevate lahendustega

Selle bakalaureustööga samal ajal valmis ka Kift platvormile veebilahendus, mille valmimist on kajastatud ühe teise tudengi lõputöös. Seega ei hakka autor välja tooma võrdlust veebilehe kättesaadavus versus rakenduse kasutatavus.

Autori arvates on Kift *client* rakenduse suurimaks eeliseks oma konkurentide ees ürituste hallatus. Olemasolevate lahenduste suurimaks probleemiks olid kaks äärmust. Esiteks toob autor välja liigse info ning oskamatuse seda kasutajale mugavalt kuvada ning teiseks äärmuseks on piiratud ehk väheste ürituste kuvamine. Autori arvates suudab Kift kasutajale pakkuda sellist lahendust, mis võimaldab avastada uusi ja huvitavaid üritusi, kuid jätab alles võimaluse leida ka kitsama kategooria üritusi.

Samuti on tähtsal kohal kasutajakogemus. Olemasolevate lahenduste üheks probleemiks selles valdkonnas oli info kuvamine tabelite ja ridadena. Selline lahendus annab küll võimaluse ühel lehel palju infot kuvada, kuid muudab kasutaja jaoks keeruliseks info leidmise.

Autor vältis *mystery-meat* navigeerimise probleemi ning valminud rakenduses üritati minimaliseerida kasutajale segadust tekitavad olukorrad.

Lisafunktsionaalsusena on Kift *client* rakenduses kaardivaade, kust on võimalik saada kiirelt ja mugavalt infot kasutaja ümber toimuvate ürituste kohta. Olemasolevates lahendustes selline funktsionaalsus puudus.

Lisaks on Kift *client* rakendusel erikategooria võimekus, mis tähendab, et kui mõni festival või üritus soovib oma ajakava rakenduses näidata, siis selle saab lisada ning seejärel kuvatakse kasutajale ka erikategooria üritused. Näiteks toob autor Tallinn Music Weeki, mille raames toimus väga palju erinevaid kontserte. Selliseid festivale toimub Tallinnas tihti ning autor usub, et nende ajakavade toomine ühte kohta võimaldab inimestel saada paremat ülevaadet toimuvatest üritustest.

3.5 Edasiarendamise suunad

Järgnevas alapeatükis toob autor välja mõned edasiarendamise võimalused, kuidas *client* rakendust kasutajale veel mugavamaks teha.

3.5.1 *Notification*-id kindlate ürituste kohta

Praegusel hetkel on rakendus võimeline vastu võtma küll kõiksuguseid *notification*-eid, kuid nendele vajutades suudab rakendus avada vaid ürituste listi vaate. Esimeseks uueks funktsionaalsuseks valis autor kindlate ürituste *notification*-id. See tähendab, et rakendus oleks võimeline kuvama kindlat üritust, mida *notification*-ga tutvustatakse. See tekitaks olukorra, kus Kift *client* rakendust saaks kasutada ürituste promomiseks.

3.5.2 Kalendri vaade

Samuti on väga tähtsaks funktsionaalsuseks ajaline filter, sest tihti on inimestel soov oma plaane mitu päeva ette teha. Praegusel hetkel rakenduses sellist võimalust ei ole. Tulevikus võiks olla võimalik avada rakenduses ka kalendri vaade, kus on võimalik täpsustada, mis kuupäevadel toimuvaid üritusi soovitakse näha.

3.5.3 Filter, et näha, mis praegusel hetkel toimub

Valminud rakenduses kuvatakse küll lisainformatsioon, kui üritus toimub lähiajal või on praegu käimas, kuid rakendusse võiks tulevikus lisada ka funktsionaalsuse, kus kasutaja saab kiiresti vaadata just praegusel hetkel toimuvaid üritusi.

3.5.4 Kategooriate edasiarendus

Praegu on rakenduses 4 kategooriat - *Kift*, *Dance*, *Sport* ja *Concert*. Samuti saab kuvada ka erikategooriat. Selline lahendus on küll toimiv, kuid autor usub, et ajapikku tuleb uurida kasutajate tagasisidet ning sellest olenevalt edasi arendada kategooriaid. Üheks näiteks võib tuua *Dance* kategooria, mis antud hetkel hõlmab kõiki pidusid, kuid tulevikus võiks sellel olla veel oma alamkategooriad, mis täpsustavad peo tüüpi.

4. Kokkuvõte

Käesoleva bakalaureusetöö eesmärkideks olid:

- Analüüsida olemasolevaid lahendusi, mis pakuvad ürituste leidmise teenust
- Luua rakendus, millega saab hallata ürituste leidmise süsteemi
- Luua rakendus, millega saab kiirelt ja mugavalt leida enda ümber toimuvaid üritusi

Töö käigus analüüsis autor olemasolevaid lahendusi ning tõi välja nende positiivsed ja negatiivsed küljed. Suurimaks miinuseks olemasolevate veebilehtede juures oli täieliku lahenduse puudumine. Mõned teenused on keskendunud liialt ühele valdkonnale, samas teiste probleemiks oli liigne info ning oskamatus seda mugavalt kasutajale kuvada. Samuti jõudis autor järeldusele, et enamus turul olevatest lehtedest on keeruka ja mitte kasutajasõbraliku disainilahendusega.

Bakalaureusetöö käigus püstitatud eesmärgid said täidetud ning töö tulemusena valmisid kaks Androidi rakendust:

- Kift *back office* rakendus
- Kift *client* rakendus

Kift *back office* rakendus võimaldab kasutajal kiirelt ja mugavalt hallata üritustega kaasnevat süsteemi. Rakenduses on võimalik määrata ürituste kategooriaid, muuta ürituse kohta käivat infot ning lisada ning hallata Kift serveris olevaid Facebooki lehti, kust üritusi saadakse.

Kift *client* rakendus võimaldab kasutajal kiirelt ja mugavalt saada infot toimuvate ürituste kohta. Rakenduses on võimalik filtreerida üritusi, saada täpsustavat informatsiooni ning vaadata üritusi kaardilt. Valminud rakenduse suurimaks eeliseks konkurentide ees on ürituste kategoriseeritus, kaardivaade ning heale kasutajakogemusele orienteeritud disain.

Kasutatud kirjandus

1. Solving the Internet's TMI (too much information) problem [WWW] http://www.digitalmarket.asia/2016/12/solving-the-internets-tmi-too-much-information-problem/

2. With 450M Users, Facebook Events Is Primed For A Standalone App [WWW] https://techcrunch.com/2015/07/29/will-facebook-launch-an-events-app/

3. Eventbrite - Fun Local Events [WWW] https://play.google.com/store/apps/details?id=com.eventbrite.attendee&hl=en

4. YPlan - Live Your City [WWW] https://play.google.com/store/apps/details?id=com.yplanapp&hl=en

5. Nearify - Discover Events [WWW] https://play.google.com/store/apps/details?id=com.nearify.android&hl=en

6. Facebook Messenger- Wikipedia, the free encyclopedia [WWW] https://en.wikipedia.org/wiki/Facebook_Messenger

7. Facebook- Wikipedia, the free encyclopedia [WWW] https://en.wikipedia.org/wiki/Facebook

8. Facebook Events app comes to Android [WWW] https://techcrunch.com/2016/12/12/facebook-events-android/

9. Introducing the Events From Facebook App [WWW] https://newsroom.fb.com/news/2016/10/introducing-the-events-from-facebook-app/

10. Events from Facebook [WWW] https://play.google.com/store/apps/details?id=com.facebook.Socal&hl=en

11. Täna Tallinn [WWW] http://t2na.ee/tallinn.php

12. Täna Tallinn Facebooki leht [WWW] https://www.facebook.com/TanaTallinn/

13. Kuhuminna.ee [WWW] http://kuhuminna.ee

14. Kuhiminna.ee kino kategooria [WWW] http://kuhuminna.ee/kino/

15. Minek.ee [WWW] https://www.minek.ee/tallinn

16. Huvi.tallinn.ee [WWW] http://huvi.tallinn.ee/

17. Latest Gartner data shows iOS vs Android battle shaping up much like Mac vs Windows[WWW]
https://9to5mac.com/2016/08/18/android-ios-smartphone-market-share/

18. Android (operating system)- Wikipedia, the free encyclopedia [WWW]
https://en.wikipedia.org/wiki/Android_(operating_system)

19. Android, the world's most popular mobile platform | Android Developers [WWW]
https://developer.android.com/about/dashboards/index.html

20. 12 great Lollipop APIs every Android 5.0 developer will love [WWW]
http://www.infoworld.com/article/2842952/mobile-development/12-great-lollipop-apis-every-android-5-0-developer-will-love.html

21. Material Design and the Mystery Meat Navigation Problem [WWW]
https://medium.freecodecamp.com/material-design-and-the-mystery-meat-navigation-problem-65425fb5b52e

22. Kotlin vs. Java: First Impressions Using Kotlin for a Commercial Android Project [WWW]
https://arctouch.com/2017/05/kotlin-vs-java/

23. Kotlin vs Java: Will Kotlin Replace Java? [WWW]
https://themindstudios.com/blog/kotlin-vs-java-will-kotlin-replace-java/

24. SimpleMVP - Github [WWW]
https://github.com/KasparPeterson/SimpleMVP

25. Model-View-Presenter in Android [WWW]
http://www.tinmegali.com/en/model-view-presenter-android-part-1/

26. Introduction to Model View Presenter on Android [WWW]
http://konmik.com/post/introduction_to_model_view_presenter_on_android/

27. RxJava – Reactive Extensions for the JVM [WWW]
https://github.com/ReactiveX/RxJava

28. Using reactive programming with RxJava 2.0 [WWW]
http://www.vogella.com/tutorials/RxJava/article.html

29. Retrofit - Github [WWW]
http://square.github.io/retrofit/

30. GSON - Github [WWW]
https://github.com/google/gson

31. Firebase Cloud Messaging [WWW]
https://firebase.google.com/docs/cloud-messaging/

32. Android Studio, The official IDE for Android [WWW]
https://developer.android.com/studio/index.html

33. Postman [WWW]
https://www.getpostman.com/

34. RxKotlin & RxAndroid- Github [WWW]
https://github.com/ReactiveX/RxKotlin

35. Chrisjenx:calligraphy- Github [WWW]
https://github.com/chrisjenx/Calligraphy

36. Picasso - Github [WWW]
https://github.com/square/picasso

37. Lottie- Github [WWW]
https://github.com/airbnb/lottie-android

38. Grantland, AutofitTextView - Github [WWW]
https://github.com/grantland/android-autofittextview

39. Google Maps API [WWW]
https://developers.google.com/maps/documentation/android-api/start

40. Google Play [WWW]
https://en.wikipedia.org/wiki/Google_Play

41. Application programming interface [WWW]
https://en.wikipedia.org/wiki/Application_programming_interface

42. Ripple Drawable [WWW]
https://developer.android.com/reference/android/graphics/drawable/RippleDrawable.html

43. Hamburger menu [WWW]
https://en.wikipedia.org/wiki/Hamburger_button

44. Facebook Graph API [WWW]
https://developers.facebook.com/docs/graph-api

45. JSON [WWW]
https://en.wikipedia.org/wiki/JSON

46. Android toast [WWW]
https://developer.android.com/reference/android/widget/Toast.ht