TALLINNA TEHNIKAKÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Mobiilirakenduse API

Bakalaurusetöö

Üliõpilane: Rait Rohi
Üliõpilaskood: 112140
Juhendaja: Raul Liivrand

Tallinn
2015

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.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Töö eesmärk on luua mobiilirakendusele API ning tekitada võimalus ka välister arendajatel kasutada antud APIt oma rakenduste arendamisel AddGoalsi platvormile.

Töö olulisemad käsitletavad probleemid on praktilise lahenduse loomine teoreetiliste teadmiste baasilt.

Töös kirjeldatav API on reaalselt kasutatav ning on otseselt lisatud ka rakenduse arendusse.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 8 peatükki, 9 joonist, 11 tabelit.

Abstract

The aim of the work is to create an API for a mobile application and let other developers use that API for building their applications on AddGoals platform.

Main problems of the work were practical creation of the API using theoretical knowledge in a real world setting.

The current design of the API is used in an mobile application and is used as a basis to continue the work even further

The thesis is in Estonian and contains 46 pages of text, 8 chapters, 9 figures, 11 tables.

Lühendite ja mõistete sõnastik

API ***API(Application programming interface)***

Reeglistik olemasoleva valmisprogrammiga suhtlemiseks.

See võimaldab programmeerijatel kirjutada lisa- ja abiprogramme, mis täiendavad või laiendavad programmi(de) funktsionaalsust. [1]

Mobiilirakendus ***Mobile Application***

Nutitelefonidele, tahvelarvutitele, nuhvlitele ja muudele mobiilseadmetele mõeldud tarkvara. [2]

Beacon ***Beacon***

Bluetooth 4 põhinev asukohta kinnitav majakas, mis saab saata sõnumeid Bluetooth 4 ühinduvatele telefonidele [3]

IMEI ***IMEI(International Mobile Equipment Identity)***

Mobiiltelefoni unikaalne seerianumber. Telefoni IMEI-kood identifitseerib üheselt konkreetse seadme (sh. seadme mudeli). [4]

Autentimine ***Authentication***

Kinnituse andmine olemi väidetava identiteedi, tunnusomaduse või päritolu õigsusele; protsess, millega üks kasutaja, süsteem vm olem saab kontrollida teise olemi väidetava identiteedi tõesust, tavaliselt mingi esitatud spetsiifilise teabe (näiteks parooli), eseme (näiteks kiipkaardi vm turvatõendi) või eristava püsitunnuse (biomeetriku) alusel. [5]

Platvorm ***Platform***

Infotehnoloogias arvuti riistvara arhitektuur ja tarkvaraline baas, mis koos võimaldavad kasutada rakendustarkvara

REST	***REST*** Arhitektuuri stiil võrkudevaheliste rakenduste disainiks.
JSON	***JSON*** andmevahetus formaat, mis võimaldab inimloetavalt kirjeldada objekte ja teisi vastavaid andmestruktuure.
FILESTREAM	***FILESTREAM*** SQL serverile omane andmetüüp, mis lubab otse andmebaasi salvestada suuri faile. [6]
MIME TÜÜP	***MIME type*** ‘Interneti e-POSTi laiendused’ (inglise keeles Multipurpose Internet Mail Extensions). See võimaldab arvutil määrata failitüübi ilma iga faili vormingu analüüsimiseks eraldi avamata. [6]
ORM	***ORM(Object-relational mapping)*** Objekt-relatsiooniline kaardistamine erinevate keelte vahel
HTTP	***HTTP*** Võrguprotokoll, mis aitab veebilehitsejatel serveritega suhelda veebis.

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Hästi ehitatud API aitab nii arendajat kui lõppkasutajat. Viimaste aastatega on välja kujunenud erinevaid viise APIde kirjutamiseks. Selle lõPUTöö eesmärk on uurida API kirjutamisviisi ja kuidas API-ga lahendada erinevaid tarkvaralisi probleeme. Näitena kasutan AddGoalsi poolt loodavat aardejahtimismobiilirakendust millele ja kõik näited on seotud reaalselt arenduse käigus tekkinud küsimustega ja nende lahendamisega.

1.1 Taust ja probleem

Tehnoloogiat täis maailmas on tähtis erinevate komponentide vaheline infovahetus. Kuidas seda infot jagada nii, et seda oleks võimalik kasutada ka siis kui süsteem laieneb.
Töö on vajalik, et selgitada kuidas hoida süsteem skaleeruv. Töö on vajalik, et aidata erinevatel osapooltel asjadest ühte moodi aru saada.
Lõputöö on koostatud 2015. aasta kevadel. Töö tegemisel on võetud aluseks reaalselt ehitatav mobiilirakendus.

1.2 Ülesande püstitus

Lõputöö eesmärgid on:
Eesmärk 1: Uurida kuidas APIga on võimalik lahendada erinevaid projekti spetsiifilisi probleeme
Eesmärk 2: Uurida kuidas kirjutada APIt süsteemi sisemisele arendusele ja arendajatele, kes hakkavad seda APIt kasutama oma rakenduste kirjutamiseks

1.3 Metoodika

Eesmärgini 1 jõuan vastavate probleemide kerkimisel mobiilirakenduse kirjutamisel. Eesmärk 2 laheneb koostöös mobiilirakenduse laiendatud meeskonnaga, mis hõlmab endas nii arendajaid kui ka müügiinimesi.

1.4 Ülevaade tööst

Kõigepealt annan kirjelduse loodavast mobiilirakendusest. Seejärel kirjeldan funktsionaalsusi, mida on vaja lahendada läbi API. Iga funktsionaalsuse juures toon erinevad võimalused probleemi lahenduseks, kasutades reaalseid näiteid seal, kus võimalik ning eraldi tõstan esile meetodi, mida kasutan ja miks. Kokkuvõttes üldistan ning toon järeldusi API kirjutamiste võimalustest.

2. Mobiilirakenduse kirjeldus

Mobiilirakendus luuakse AddGoalsi platvormi toetavale rakendusele. AddGoals on platvorm, mis lubab inimestel jagada ning leida eesmärke ning tegevusi, mida nad sooviksid elu jooksul ära teha. Mobiilirakenduse lisaks AddGoalsile veidi mängulisust ning oleks üks võimalik turunduskanal.

2.1 Mis probleemi lahendab?

Mobiilirakendusel on kaks suuremat eesmärki. Esimene on toetada väljakutsete lisamise platvormi AddGoals rakendusega, mis lubaks kasutajatel täita eelnevalt täidetud nimekirja väljakutsetest ning läbi selle teha toodet atraktiivsemaks. Teine eesmärk on AddGoalsi klientidele pakkuda lisavõimalusi oma toote või teenuse turule panemiseks.
Mobiilirakenduses on kolm suuremat osa:

1. Eelnevalt kokku pandud väljakutsete nimekirjade sirvimine.
    a. AddGoalsi platvormil on kokku pandud nimekirjad erinevatest tegevustest ning lühitutvustustest, mida pakett hõlmab edaspidi tuntud kui *event*. *Event* on esindatud lühitutvustuse, pildi, punktide arvu ning ka maksumusega. Igal *eventil* on mitu alampunkti, mida tuleb täita edaspidi tuntud kui *eventitem*.

2. Kasutaja eventide haldamine. Mobiilirakenduses peab olema võimalik näha kasutajaga seotud evente ning ka kasutajaga siduda evente vastavate reeglitle järgi.

3. Kasutaja konkreetse *event*i *eventitem*ite täitmise kontrollimine.

    a. *eventitem*eid kontrollitakse kolmel viisil:

        i. Pildi või videoga. Mobiilirakenduse siseselt kontrollitakse kas foto või video on tehtud ja salvestatud.

        ii. Asukohaga. Asukohti kontrollitakse kas telefoni GPSi või Beacon abil.

        iii. Küsimusega. Kasutajale esitatakse küsimus, mida kontrollitakse.

    b. Kui kõik *eventitem*id on täidetud või siis piisavas Eventis määratud punktide arv kogutud on kasutajal võimalik event sulgeda. Eventi sulgemine tähendab täidetud punktide, meedia ning vastuste edastamist serverile.

Mobiilirakenduse eesmärkide täitmiseks on vaja korraliku APIt, mis lubaks antud infot edastada ning talletada.

2.2 API funktsionaalsus ja kirjeldus

API peab täitma kolme suuremat eesmärki.

1. Äriloogikaga seotud ja eeldab seda, et peale eventi allalaadimist ei ole tarvis enam serveriga suhelda. See tähendab, et eelnevalt peavad olema edastatud kõik *eventitemi*ga seotud info ning kontrollmehhanismid.
2. Pakkuda äriklientidele võimalus luua ning ka hallata evente. See tähendab, et APIs on ka osa mida ei kasutata otseselt mobiilirakenduses.
3. Eelneva peatüki lõpus mainitud info edastamine ning haldamine.

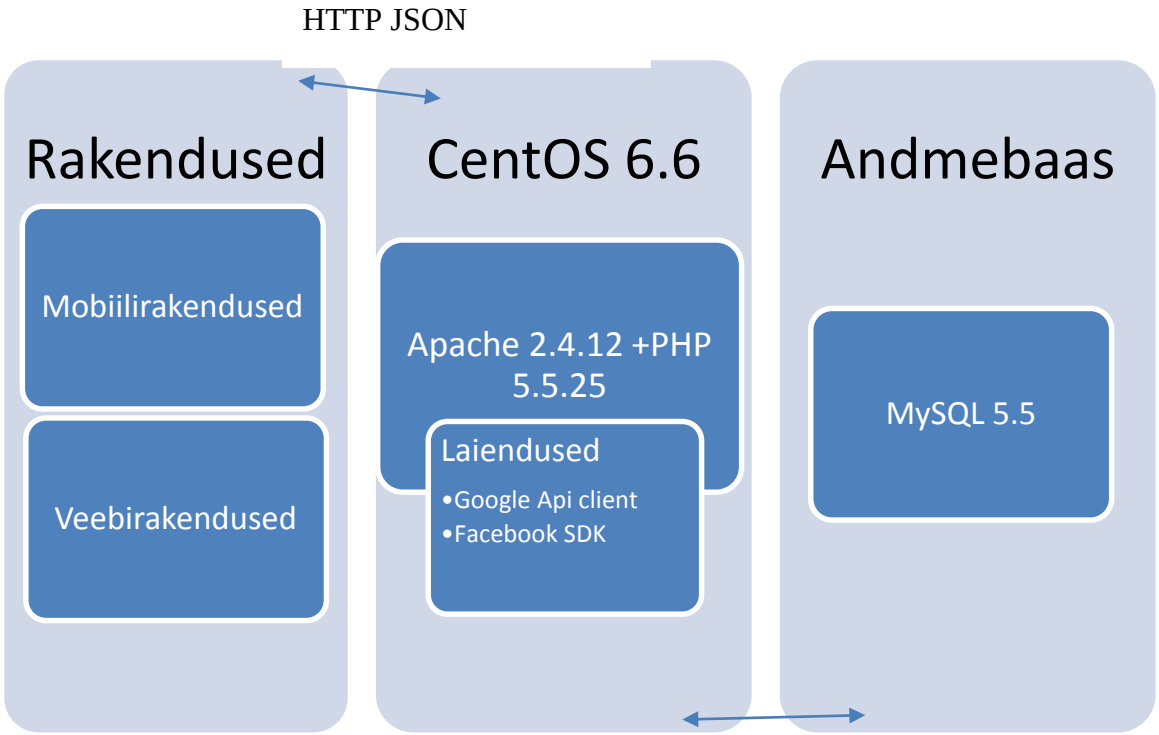
API kirjutamisel pean silmas mõningaid põhimõtteid:

1. Minimalism. Kõik kutsed peavad olema võimalikult lihtsad ja sisaldama endas võimalikult avalikke klasse. Tähtis parem arusaamine, korrigeerimine ja API kutsete mäletamine.
2. API peab täitma kogu funktsionaalsust.
3. Selge semantika. Kutsed peaksid järgime ühtset loogikat.
4. Intuitiivsus. Kutse peaks tegema seda mida arendaja arvab, et see teeb. Mida vähem on vaja dokumentatsioonist lugeda seda parem.

APIga peab olema realiseeritud ka kasutajate autentimine. See on vajalik nii mobiilirakenduse kui ka AddGoalsi klientide sisestatud eventide jaoks. Tähtis on, et seda oleks võimalikult lihtne teha ning ei nõuaks palju aega.

3. Üldine metoodika

Üldise metoodika all kirjeldan kõigi kutsete kirjutamise viise vastavalt sellele, mida kutse peab täitma. Toon ka realiseeritud näiteid. Põhiline on, et mobiilirakenduseks vajalikud kutsed saaks realiseeritud ning lisana välistel arendajatel oleks võimalik selgelt aru saada kuidas kätte saada neid huvitavat infot.



Joonis 1: Arhitektuur ja tehnoloogia

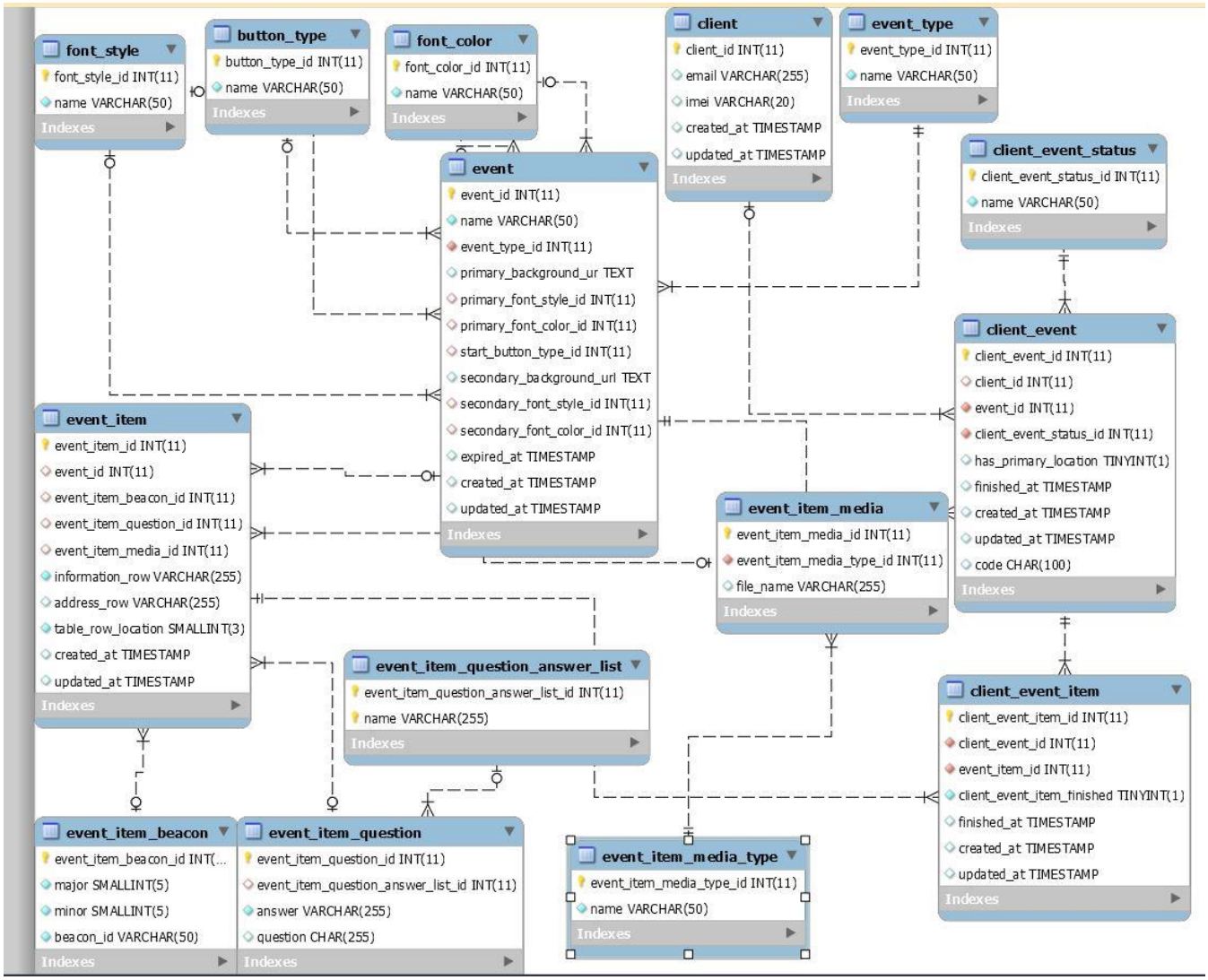
3.1 Andmebaasi kirjeldus

Andmebaasi loomisel tuli arvesse võtta juba olemasolevat infot ja disaini. Parema migratsiooni tagamiseks said nimetused loodud sarnaselt põhilisele baasile. Keeleks on MySQL, mis on samuti AddGoalsi platvormile sarnane. Tabeli primaarvõtmeks on tabeli nimi koos _id-ga. Selline disain teeb keerulisemaks ORMi valiku kuna mitmed kergemad ORMid eeldavad primaarvõtmeks ID-d.

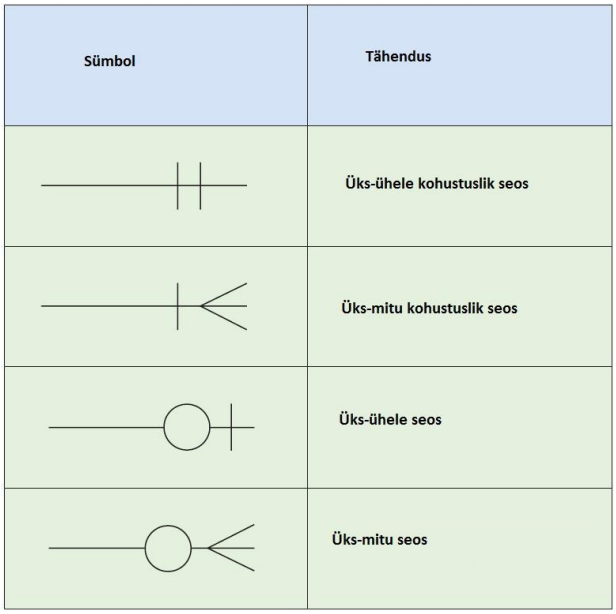
Põhilised tabelid on client, event ja event_item. Nende peale on ehitatud üles nii API kui ka aplikatsioon. Veidi halva disainina on tabelitega tihti ka seotud neid ühendavad stiilielemendid. Need said lisatud põhiliselt selle pärast, et on soov luua keskkond, kus oleks võimalik oma *event*e lisada ning hallata nende disaini mobiiliaplikatsioonis. Seetõttu on loodud ka tabelid

nagu „font_style“, „button_type“ ja „font_color“, mis iseenesest ei ütle midagi aga mängivad rolli aplikatsiooni kuvamises.

Ühel *event*il on mitu *eventitem*it. Ühel kliendil võib olla mitu *event*i ja neid hoitakse client_event vahetabelis. *Eventitem*i kontrollimeetodid ja vastused on omaette kolmes tabelis event_item_beacon, event_item_question ja event_item_media.



Joonis 2: Andmebaasi skeem



Joonis 3: Skeemi legend

Tabel 1: Andmebaasi tabelid

Tabelid	Seletus
client	Teenuse kasutaja
client_id	Kasutaja ID
email	Email
imei	Ainult mobiilselt seadmelt. Unikaalne näitaja
created_at	Loomise aeg
updated_at	Uuendamise aeg
Event	Teenus
event_id	Teenuse id
name	Teenuse nimi
event_type_id	FK, teenuse tüüp
primary_background_url	Esimesel ekraanil kasutatav taustapilt, selle url
primary_font_style_id	Esimesel ekraanil kasutatav kirjatüübi stiil
primary_font_color_id	Esimesel ekraanil kasutatav kirjatüübi värv
start_button_type_id	Alustamise nuppu kuju. Mobiilsel seadmel seadistatav
secondary_background_url	Teise ekraani taustapilt
secondary_font_style_id	Teisel ekraanil kasutatav kirjatüübi stiil
secondary_font_color_id	Teisel ekraanil kasutatav kirjatüübi värv
expired_at	Kehtivuse kaotamise aeg. Peale seda enam ei kuvata
created_at	Loomise aeg
updated_at	Uuendamise aeg
event_item	Ülesanne ehk *eventitem*
event_item_id	Ülesande ID
event_id	Teenus, mille alla kuulub
event_item_beacon_id	Asukoha määratlusega *eventitemil*. Kui määratud siis kontrollib ka asukohta.
event_item_question_id	Küsimuse ID. Kui on määratud toimub ka küsimuse kontroll

event_item_media_id	Meedia ID. Kui on määratud siis kontrollib ka meedia asukohta
information_row	Informatsioon ülesande kohta. Kuvatakse kui ülesanne avatakse
address_row	Aadressi reale kuvatav informatsioon
table_row_location	Kui on määratud siis selle järgi näidatakse kuvamise järjekorda
created_at	Loomise aeg
updated_at	Uuendamise aeg
client_event	Kliendiga seotud *event*id
client_event_id	Unikaalne näitaja
client_id	Kliendi ID
event_id	Teenuse e. *event*i id
client_event_status_id	*Event*i staatus (lõpetatud, pooleli, ostetud)
has_primary_location	Kui tõene siis selle *event*i alustamiseks peab minema algpunkti
finished_at	Lõpetamise aeg
created_at	Loomise aeg
updated_at	Uuendamise aeg
code	Unikaalne kood sellele eventile. Vajalik *event*iga sidumiseks ja *eventitem*ite kuvamiseks
client_event_item	kliendiga seotud *eventitem*id
client_event_item_id	Unikaalne muutuja
client_event_id	Kliendiga seotud *event*i ID
event_item_id	*Eventitem*i id
client_even_item_finished	Näitab kas selle *eventitem*i kontrolltingimused on täidetud
finished_at	Lõpetamise aeg
created_at	Loomise aeg
updated_at	Uuendamise aeg
event_type	*event*i tüübid
event_type_id	Unikaalne muutuja
Name	*Event* tüübi nimi. Näiteks „Suveüritused“
client_event_status	staatuste nimetused
client_event_status_id	Unikaalne muutuja
Name	Muutuja nimetus, näited toodud eelnevalt
event_item_beacon	Asukohaga kontrollitav *eventitem*
event_item_beacon_id	Unikaalne muutuja
Major	iBeaconile iseloomulik muutuja, mille järgi võimalik määrata unikaalne Beacon
Minor	iBeaconile iseloomulik muutuja, mille järgi võimalik määrata unikaalne Beacon
beacon_id	iBeaconi id. Võimalik, et mitu iBeaconit on sama ID-ga
event_item_question	Küsimusega kontrollitav *eventitem*

event_item_question_id	Unikaalne muutja
event_item_question_answer _list_id	Kui küsimus on määratud nimekirjaga siis siin viide sellele nimekirjale
Answer	Õige vastus küsimusele
question	Küsimus teksti kujul
event_item_media	Meediaga kontrollitav *eventitem*
event_item_media_id	Unikaalne muutuja
event_item_media_type_id	Näitab, mis tüüpi fail on (audio, foto, video)
file_name	Nimetus, millega fail serverisse salvestatakse
event_item_media_type	Meedia tüübid (audio, foto, video)
event_item_media_type_id	Unikaalne muutuja
Name	Tüübi nimi
event_item_question_answer _list	Vastuste nimekiri, komaga eraldatud väärtused
event_item_question_answer _list_id	Unikaalne muutuja
Name	Komaga eraldatud vastusevariandid
font_style	Kirjatüüpide kogumik
font_style_id	PK
Name	Kirjatüübi nimetus
button_type	Nupu tüüpide kogumik
button_type_id	Unikaalne muutuja
Name	Nupu nimi
font_color	Kirjatüübi värvide kogumik
font_color_id	PK
Name	Kirjatüübi värv

3.2 REST põhimõtted

API disainis lähtusin mõistlikkuse piires RESTful API põhimõtetest. See tähendab, et ei päringud on üksteisest sõltumatud. GET päringute põhjal kasutatakse cache, et kiirendada päringuid. Kõik päringud peavad olema standard HTTP meetodite järgi- PUT, GET, POST, DELETE.

3.3 Haldamise REST meetodid

Enamus API loogikat põhineb andmebaasist CRUD päringute realiseerimisele. Kirjeldan siin kuidas see on konkreetses APIs ja miks. Read päringud saavutatakse GET abil. Kui tulemus tagastab kindlasti ainult ühe objekti siis päring näeb välja nii: <Objekt>/<Objekti id>. Kui tagastatakse kas üks või mitu objekti siis päring näeb välja nii: <Objekt>+s

Tabel 2: Mitmuse ja ainsuse kutse

Meetod	Kutse	Kirjeldus tagastusele
GET	/Event/1	Tagastab eventi, mille id on 1
GET	/EventItems/1	Tagastab kõik eventiga, milled id on 1 seotud tegevused.

Delete kasutatakse API siseselt vähe äriloogika nõuab info säilitamist. Vähesed kustutuskutsed on realiseeritud DELETE meetodi abil ja vajalik on sessioni id. Üldine kustutamine on staatuse muutmine ja realiseeritud PUT päringu abil. Lihtsamad näited :

Tabel 2: Kustutamine

Meetod	Kutse	Kirjeldus tagastusele	POST parameetrid
DELETE	/Event/1/7f4f9920-e20e-11e4-a65b-0002a5d5c51b	Kustutab eventi id-ga 1	
PUT	/Client/1	Tagastab kõik eventiga, mille id on 1 seotud tegevused.	{status: ’DELETED’}

Create ja Update realiseeritakse mõlemad POST meetodi abil. Vahet tehakse valikulise id parameetriga. Create päring on kujul <Loodava objekti nimi>/<sessiooni number>.Update päring on kujul /<Loodav objekti nimi>/<Objekti id>/<sessiooni number>. Kui see on olemas siis tehakse update lause.

Tabel 3: Objekti loomine ja uuendamine

Meetod	Kutse	Kirjeldus tagastusele	POST parameetrid
POST	/Client/7f4f9920-e20e-11e4-a65b-0002a5d5c51b	Loob uue kliendi	{name:'Test1', Email:'test@test.ee' }
PUT	/Client/1/ 7f4f9920-e20e-11e4-a65b-0002a5d5c51b	Muudab kliendil id-ga 1 vastavaid parameetreid	{name: 'Test2'}

3.4 Filtrid ja sorteerimine

Alati ei piisa sellest, et tagastatakse kogu tabelites olev info. Otsus filtreerida API siseselt mitte mobiilirakenduses tuleneb minimalismi põhimõttest. Samuti aitab see vähesel määral vähendada ka andmemahtu. Filtreerimise põhimõttel toimivad ka kõik otsingud.

Sorteerimise küsimus tekib siis kui andmeid on palju ning kogu andmejada tuleb ka kasutajale kuvada. See tuleneb suuremalt osalt siis kui *eventItem*eid on palju. Juba andmebaasi struktuurist on näha, et *eventitem*itele on määratud ka positsiooni väärtus. Sorteerimine on esialgses lahenduses madalaprioriteetsel kohal, kuid tuleviku visiooni kohaselt on selle realiseerimine vajalik ka API poolelt

Tabel 4: Tüübiseletused

Nimetus	Selgitus
String	Tavaline sõna või mumber
Char	Üks täht või number
Int	Täisarv
Date	Kuupäev

3.4.1 Filtrid

Esimene otsus filtri loomisel on kuidas seda välja kutsuda ning siis edaspidi kasutada sama loogikat igas kutses. Välja pakutud lahendus oleks luua filter JSON massiiv, milles nimi annaks teada, mille järgi sorteerida ning väärtus siis vastavalt, milline peaks olema oodatav vastus. Siinkohal tekivad küsimused, millele veel lahendust vaja:

1. Filtreering kui ei tea nime väärtust ehk kõikehõlmav otsing
2. Mittetäielik otsing, ehk otsing ei vasta täielikult leitavale osale
3. Keerulisemate ehk täpsemate otsingute realiseerimine

Kõikehõlmava otsingu jaoks peab API siseselt määrama tabeli väljad, mille järgi otsitakse kui filtri nimeväli on 'ALL'.

Tabel 5: Filtreerimine

Päringu näide	Kutse	Selgitus	Näitevastus
{filter:{ „ALL“:[„test“ , „jaan“, „tegus“] }}	EventItems/2	Otsib tabelist eventitem kõiki eventitemeid, mis sisaldavad kas sõna „test“ või „jaan“ või „tegus“	{ eventitem : [{„id“ : 1, „name“ : „Tegus Jaan“}, {„id“ : 2, „name“: „Test event“ }],

Mittetäielik otsing realiseeritakse abimuutujatega. Kui otsitav sõna ehk string algab või lõpeb „%“ tähemärgiga siis API jaoks tähendab, see otsimist kõikidele stringidele, mis on sarnased päringule. Näitena otsing „%test“ leiaks ka stringid „katsetest“ , „test“. Mobiilirakenduse otsingutes pannakse iga uue stringi algusesse „%“. See tähendab, et otsing „Katse Otsing“

annaks APIle otsingusõnad „%katse%“ ja „%otsing%“. Tulevikus on soov täiustada otsingumootorit, et tulemus oleks lähedasem sellele mida soovitakse leida.

Keerulisemad otsingud jäävad mobiilirakendusest endast välja kuid on siiski realiseeritud võimaliku tulevikuvajaduse tõttu. Selle jaoks on filtris kasutuses veel selline nime märksõna nagu „operation“. Kui vastavat väärtust pole võimalik otsitava välja juures kasutada siis ignoreeritakse seda. Andes „operation“ mitu väärtust siis antakse vastuseks veateade. Sellele aktsepteeritakse hulk väärtuseid :

Tabel 6: Filtreerimise võimalused

Väärtused	Selgitus	Tabeli väärtuste väljad, millel võimalik kasutada
>, gt	Suurem kui.	Date, int
<, lt	Väiksem kui	Date, int
>=, gte	Suurem või võrdne	Date, int
<=, lte	Väiksem või võrdne	Date, int
In	Vastusesse kuulub üks edastatava massiivi osa	String, Char, int
not_in	Vastusesse ei kuulu ükski edastatava massiivi osa	String, Char, int

3.4.2 Sorteerimine

Sorteerimine on samuti realiseeritud tulevikuvajaduse tõttu. Mobiilirakenduse siseselt sorteerimist ei kasutata. Samas API lubab seda teha kasutades märksõna „sort“. Sort parameetri kohustuslik nimiväärtus on „name“ ja eeldab vastavat välja, mille järgi sorteeritakse. Kui see on ainus parameeter siis tehakse vaikesorteering, milleks on siis vastavalt kas suurus (int, date väljadel) või tähestikuline (string, char väljadel). Names lubatakse ka massiivi, mille puhul on oluline ka järjestatus. See tähendab, et kui nimeväljas on rohkem kui üks parameeter siis algselt

sorteeritakse esimese parameetri järgi ning kui see on vastuses sama siis sorteeritakse järgmise parameetri järgi. Nimekiri parameetritest ja väärtuste väljadest, mida sort aksepteerib:

Tabel 7: Sorteerimine

Nimi	Väärtused	Seletus
dir	ASC, DESC	Kas sorteerimine suuremast väiksemaks või vastupidi
limit	Täisarv	Näitab vastavat täisarv tulemusi

3.5 Nimekiri põhilistest kutsetest

Tabel 8: Põhilised API kutsed

Kutse	Mida teeb?	Seletus	Mille jaoks vajalik
/events/	Tagastab kõik praegusel hetkel veel aktiivsed üritused, mida võimalik kuvada	Admin poole peal võimalik määrata järjestust kategooriate järgi. Eelnevalt nimetatud filtrid ja sorteerimine võimalik kasutada ka lõppkasutajal	Näidata kliendile kõiki Evente, mida saaks läbida. Kasulik nii avastamiseks kui uute eventide saamiseks
/events/token	Tagastab kõik kliendiga seotud eventid. Neid evente on reaalselt võimalik kohe hakata läbima	Põhimõtteliselt on see kliendiga filtreeritud /events/ kutse	Näidata kliendile neid evente, mis ta on endaga juba sidunud

/event/token	Seob antud *event*i kliendiga	Luuakse uus kirje tabelisse ja tekivad seosed. Post parameetriga kaasa kirje *event*i omamise tõestamiseks	Kliendile uute *event*ide pakkumiseks
/eventitems/ <eventid>/ <token>	Kuvab kliendile kõik valitud eventiga seotud *item*id	Siinkohal tuleb kaasa ka eventiga spetsiifiline info nagu taust, fontid jms. Vajalik ärianalüüsist tulenevast nõudest töötamisest ka väljaspool võrku.	Nimekirja kuvamiseks koos staatustega
/eventitem / <eventid> / <itemid> / <token>	Tagastab konkreetse *item*i, ning sellega seotud info koos staatusega. PUT meetodil võimalik muuta konkreetse itemi staatust	GET meetodid ei kasutata üldse kui pole internetineti ühendust.	Vajalik välistele arendajatele ja tagastaks ainult üh
/eventitems/ <eventid>/ <token> POST meetodil	Eraldi väljatoodud kuna spetsiifiline rakendusele ja kõige vearohkem protsess	Kui lõppkasutaja on eventiga lõpule jõudnud ja nüüd tagasi võrgus, on vaja uuendada kõiki itemeid. Selle jaoks salvestatakse	Tähtis selle jaoks, et oleks võimalik tehtud muutused korraga ette sööta. Kui token on selle ajaga aegunud tuleb uuesti teha login.

		pidevalt infot ning edastatakse see siis ühe suure POST päringuga serverile.	See on kogu API üks spetsiifilisemaid ja veaohtlikumaid kohti. Siia sisse on kirjutatud ka mitmeid vigade parandamise kohti
/nearbyevent/<token> POST meetodil	Tagastab kasutajale läheduses olevad *event*id. See on seotud eventi koordinaatidega	Kasutatakse siis kui eventiga on seotud andmebaasis ka koordinaadid. POSTiga peavad kaasas olema praegused koordinaadid	Liikvel inimestele soovitada lähedal asuvaid *evente*.
/validate/<eventitemid>/<token>	Kontrollib, kas vastus *eventitem*ile oli korrektne	POST parameetritega kaasas *eventitem*ile pakutud vastus. Tagastab kas see oli tõene või väär	Vajalik välistel arendajatel vastuste kontrollimiseks.
/style/<eventid>/<token>	Tagastab kliendi *event*iga seotud stiilid	Edasiarendusena mõeldud osa. Eesmärgiks pakkuda erinevaid viise *evente* kuvada mobiiliseadmetel	Vajalik kasutajale kasutajakogemuse kujundamiseks
/media/<eventitemid>/<token>	Tagastab antuda *eventitem*iga seotud meedia	Eeldab, et on täidetud kaks tingimust: 1) Vastuseks on vaja	Välistele arendajatele tõestusmaterjali saamiseks mingi

		kasutada meediat (pilti, video, lindistust) 2) Pilt on tehtud ja eksisteerib	*eventitem*i täitmisest. Kui selle kohta meediat pole annab kutsujale ka seda teada
Kasutaja gruppidega seotud kutsed (Tulevikus realiseerimiseks)			
/friend/<token>	Tagastab kasutajaga seotud sõbrad		Annab välistele arendajatele võimaluse saada kätte kasutaja sõbrad.
/friend/event/<token>	Tagastab kasutaja sõprade *event*id		Annab välistele kasutajatele võimaluse näha sõprade üritus
/friend/<id>/<token>	Tagastab kasutajale konkreetse sõbra profiili	Tagastatakse ainult info, mis ei ole privaatne. Sisemine kontroll, kas antud tokeniga seotud kasutaja on sõbra staatuses valitud kasutajaga	Võimalus näha konkreetse sõbra profiili. Eeldab, et teada on ka sõbra id
/setting/<token>	Lubab muuta vaikesätteid vastavalt parameetritele	Võimalik muuta enda vaike sorteeringut ja näidatavat infot	Seotud kasutajakogemuse mugavamaks muutmisega

/challenge/ <friendid> /<token>	Saadab sõbrale kutse teha valitud *event*i või *eventitem*it	Post parameetritega kaasas kas *event* või *eventitem*	Eesmärgiks parem jagamise võimalus rakenduste siseselt.
***Event*i sotsiaalse poolega seotud kutsed (Tulevikus realiseerimiseks)**			
/eventcom/ <eventid>/<token>	Kommentaari lisamiseks valitud *event*ile		Mõeldud nii rakenduse siseselt kui ka välistele arendajatele kommentaaride lisamiseks *event*ile
/eventitemcom/ <eventitemid>/ <token>	Kommentaari lisamiseks valitud *eventitem*ile		Mõeldud nii rakenduse siseselt kui ka välistele arendajatele kommentaaride lisamiseks *eventitemile*
/eventrate/ <eventid>/<token>	Võimalik anda hinnang *event*ile	Süsteem aksepteerib väärtusi 1-10 ja kuvab neid 5 tähekesega	Laiem eesmärk filtreerida populaarsemaid *event*e ja anda kasutajale rohkem võimalus sorteerimiseks
/eventitemrate/ <eventitemid> / <token>	Võimalik anda hinnang *event*ile	Süsteem aksepteerib väärtusi 1-10 ja kuvab neid 5 tähekesega	Laiem eesmärk filtreerida populaarsemaid *eventitem*eid ja anda kasutajale rohkem

			võimalus sorteerimiseks

4. Kasutaja sisselogimine

Igasuguse rakenduse või veebilahenduse tähtis osa on kasutajate haldamine. Antud mobiilirakenduse veebirakenduses on realiseeritud autentimine läbi Google+ ja Facebooki. Tuleviku mõttes on soov liita ka võimalus kasutajanime ning parooli kasutamiseks. Lahenduse teeb keerukamaks veel ärianalüüsist tulenev soov lubada kasutajal kasutada rakendust ilma sisse logimata ning siis hiljem enda tehtud ülesandeid siduda oma veebikontoga. See tähendab, et vaja on salvestada ka IMEI kontod.

4.1 Valitud lahendused ja API kutsed

Realiseeritud on kõik 4 kasutaja logimist. Lahenduste keerukam pool on otseselt mobiilirakenduses kuid osa keerukust langeb ka kahekordsele autentimisele. Kasutaja sisselogimisest oma kontole on vaja teavitada ka serverit. Mõlema puhul käib veel sisemine kontroll tokenile ehk sisuliselt on tegemist kahekordse logimisega. Kõigil kontodel peale IMEI on ärilises mõistes üks ja tähtis siduv osa: e-mail. E-mailiga kontrollitakse ka kasutaja unikaalsust.

4.1.1 Facebookiga ja Google+ autentimine

Facebookiga sisselogimisel peab Facebooki käest küsima tokenit. Kuna meie huvi ei ole ainult sisselogimise vastu, vaid meil on vaja ka teada saada kasutaja meilikontot. Kui Facebook on kasutajale edastanud tokeni siis edastab kasutaja selle API kaudu serverile. Otsene kutse on:

/loginfb?token=<Facebookilt saadud token>

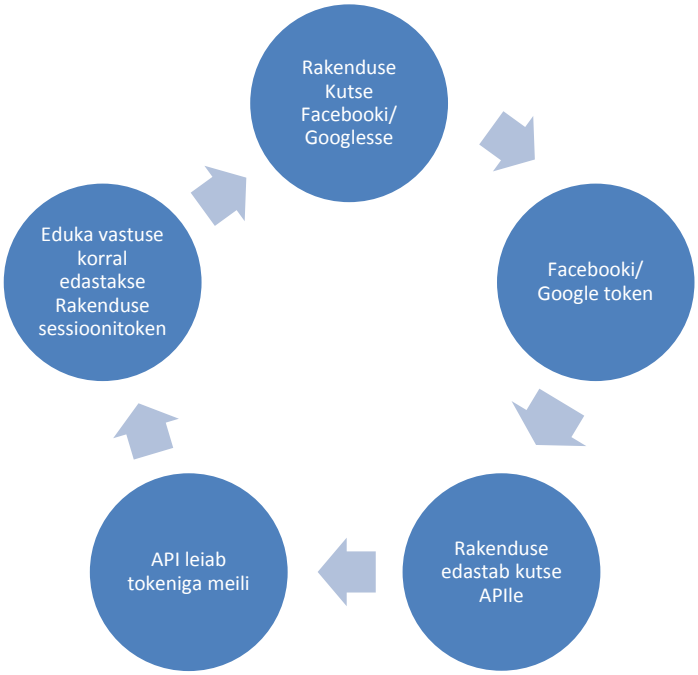
API poolelt võetakse token vastu ja tehakse päring uuesti FB serverile. Päring FB serverile näeb välja selline:

v2.3/me?access_token=<Facebookilt saadud token>&fields=email&format=json&method=GET

Saadud vastuses on kasutaja e-mail ja id. Emaili kasutaksegi kontoga sidumiseks ja kasutajale autentitaksegi

Kasutaja jaoks käib kõik ühes jadas, aga tegelikkuses on liiklus keerulisem. Facebook eristab pikajalisi ning lühiajalisi tokeneid. Lühiajaliste kestvusaeg on umbes tund ja pikajalisel umbes 60 päeva. Meie vajadustele vastab lühiajaline token.

Google+ autentimine on sarnaselt Facebookile ka tokeniga ning järgneb samale skeemile. Seetõttu ei hakanud seda ka eraldi kirjeldama



Joonis 4: Facebookiga autentimine

4.1.1.1 Tugevused ja nõrkused

Facebookiga logimisel on omad head ja halvad küljed. Tugev külg on kindlasti lihtsus ja kasutajale juba harjumuspärane protsess. Teine tugev külg, mis oli ka ärianalüüsis väga määrav, on Facebooki kasutajate hulk. Unikaalseid kasutajaid oli 2014 teiseks kvartaliks umbes 1.3 miljardit. [7]

Nõrkuseks on aga sisselogimise loogika. Mida rohkem on osapooli seda rohkem on kohti, kus midagi võib valesti minna. Peab tegelema olukordadega, kus ühe osapoolega on kõik korras, aga teisega ei ole. Tähtis on, et serveri poolelt oleks võimalik jälgida, kus viga juhtus ning kasutajale samuti teada anda. Kõige raskem on lahendada juhtu, kus edastatakse õige token, aga API poolt ei saada vastust. Selle jaoks tuleb API siseselt realiseerida uuesti katsetamine 10 korda ja peale seda eeldatakse, et viga on API poolne.

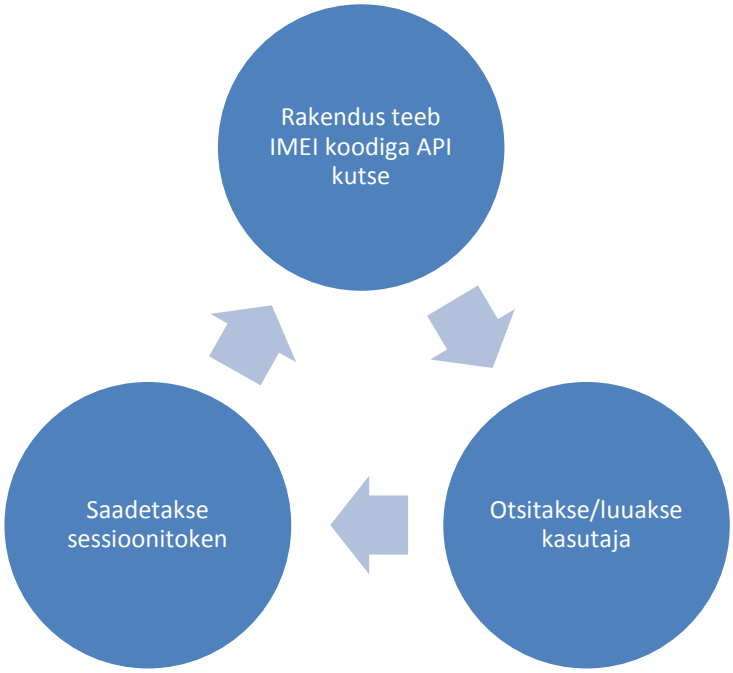
Teine riskikoht on Facebooki uuendused oma APIle. Praeguseks on Facebookil väljas API versioon 2.3. Api versiooni versiooni 1.0 toetatakse kuni 2015 aasta 30 aprillini. Üldjuhul tähendab uuendus seda, et vaja on teha muutusi nii rakenduse kui ka API siseselt. See on järjekordne lisategevus, mida tuleb hoida meeles.Lisaohte tekitab ka see kui otsustakse muuta seda, kuidas infot tagastatakse. Näiteks võib järsult tulla muutus selle kohta, missugust infot tokeniga jagatakse ja meil võib sellest infost välja jääda. Jällegi tuleks muuta loogikat mitmes kohas.

4.1.2 Kasutaja + salasõna

Kõige tavalisem kasutaja autentimise viis on salasõnaga. Selleks saadetakse APIle kasutajanimi ning salasõna ning kontrollitakse nende olemasolu andmebaasis. Edukal kontrollil luuakse access token ning seda kasutatakse järgmise päringute sooritamiseks. Päringuna kasutatakse POST parameetrit, mille sees id ja password.

4.1.3 IMEI ja mittetäielik logimine

IMEI kasutamine on suuresti seotud just sellega, et esialgne põhiline eesmärk on mobiilirakenduse loomine. Ärianalüüsi poolelt oli selge vajadus ka pakkuda sisu kasutajale, kes pole sisse loginud. Samas muutusi on soov salvestada ja võimalik, et hiljem siduda ka täielikult autentitud kontoga. Selleks saadetakse POST parameetriga APIle telefoni IMEI kood ja luuakse kasutaja, kellel on võimalik ligi saada pea kõigile API callidele, mis ei vaja tokenit. Samal ajal tehakse ka andmebaasi salvestusi vastava kasutajale. Hiljem samast telefonist kontot luues seotakse konto ja IMEI omavahel ära ning tehtud muutused ei lähe kaduma.



Joonis 5: IMEI ja mittetäielik logimine

4.1.3.1 Tekkivad probleemid ja nende lahendused

Mittetäieliku logimise lisamine tekitab keerukust API poolel. Tuleb realiseerida nii kontoga sidumine kui ka info loomine ilma tokenita. Samas tekitab see ka selgeid probleeme näiteks telefoni kasutajate vahetusel. Inimene, kes on ostnud kasutatud telefoni võib järsku näha infot, mis ei ole temaga seotud. Sellepärast on vaja luua ka API sisse funktsionaalsus andmete kustutamiseks ning staatusteks. Päriselt andmeid ei kustuta kuid määratakse staatus kustutatud ning lubatakse luua n.ö uus konto.

5. Sessiooni haldamine

API kirjutamisel on tähtis mõelda nii turvalisusele kui ka selle kasutamise mugavusele. Ei ole mõeldav, et iga API funktsiooniga, mis eeldab unikaalse kasutaja olemasolu peaks järgnema uuesti kogu sisse logimise protsess. Samas API eeldab, et iga päring on isoleeritud. Selle realiseerimiseks luuakse peale sisse logimist sessiooni-id, mida hakatakse edasi saatma igas päringus, mis eeldab unikaalset kasutajat.

5.1 Sessiooni loomine ja lõpetamine

Sessiooni loomine käib käsikäes sisselogimisega. Kasutaja edukal sisselogimisel luuakse uus sessioon, mis on seotud kasutajanimega. Vaikimisi on see sessioon aktiivne 6 tundi peale viimast päringut. See tuleneb ärianalüüsist tulenevast vajadusest rakenduse kasutamisest ilma internetita. Kui kasutajad saavad tagasi ühenduse uuendatakse seadeid ning viiakse sisse vastavat muutused. Kui vahepeal on mööda läinud üle 6 tunni peab kasutaja ennast uuesti sisselogima.

Sessiooni lõpetamine tähendab selle mitteaktiivseks muutmist. Tehnilisest küljest tähendab, et sessioon saab külge aegunud staatuse andmebaasis ja seda pole enam võimalik kasutada andmete küsimiseks.

6. Failide üles ja allalaadimine

Mobiilirakendus eeldab seda, et üles on võimalik laadida hulk pilte ja mõnikord ka teisi faile. See tekitab küsimusi kuidas neid faile kõige paremini talletada. Erinevaid viisid tekitavad ka API siseseid küsimusi. Kuidas ja isegi kas fail edastada. Valik käib kahe viisi vahel:

1. Salvestamine otse andmebaasi
2. Salvestamine serverisse eri kaustadesse

6.1 Salvestamine otse andmebaasi

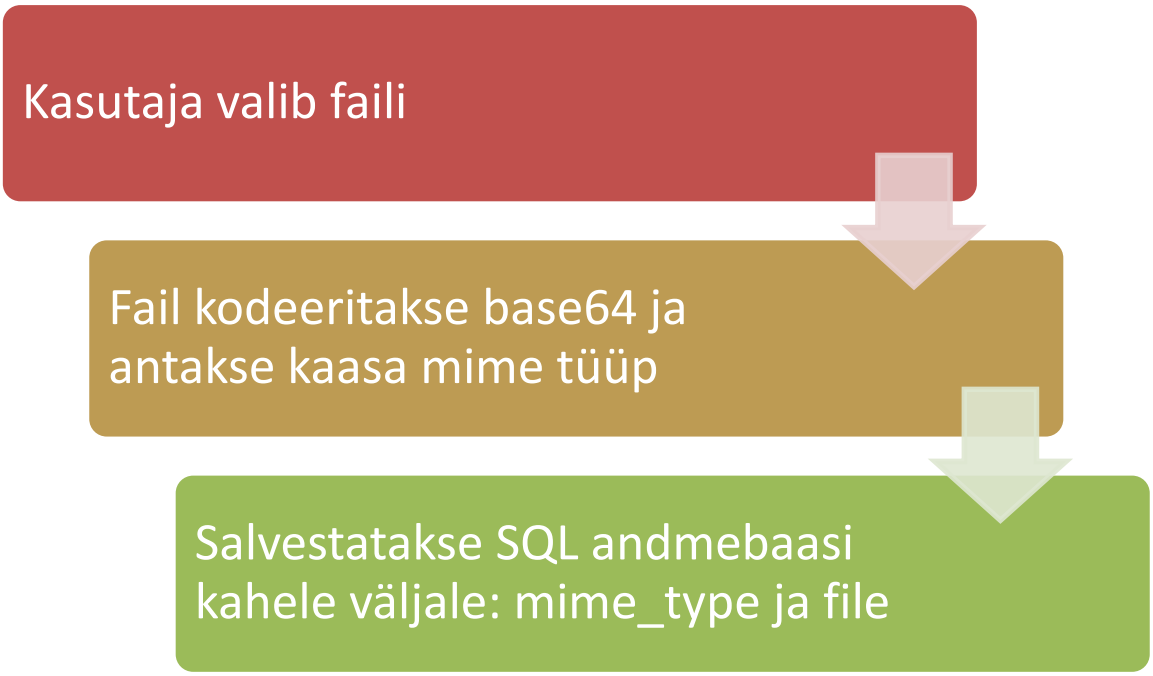
Andmebaasi salvestamise poolt räägib mitu argumenti. Esiteks on realiseeritud ACID [8]

(atomaarsus, järjepidavus, eraldatus ja püsivus), mis serveri kaustadesse salvestamine pole nii kerge. See on põhiline boonus arvestades, et andmebaas ja failid on omavahel sünkroneeritud igal ajal. Andme kadu tähendaks ka seda, et antud failile pole mingit viidet.

Lisana on kasulik ka pidev andmebaasiga seostatus. See tähendab, et failid ja andmebaas kuuluvad kogu aeg kokku. Erinevate viidete loomine on selge ja tuleneb otse andmebaasi struktuuri dokumentatsioonist ilma, et seda oleks vaja väljapoole seletada. See lihtsustab ka väljastpoolt arendajatelt failide jagamist kuna õigused tulenevad otse andmebaasist.

Poolt argumendiks tuleb ka andmete migreerimine ja ladustamine. Vaja on ühte faili ja selle taastamine ei nõua palju tööd.

Tabel 9: Faili salvestamine andmebaasi



6.2 Salvestamine serverisse

Serverisse salvestamise pooltargumendid lahendavad probleeme, mis tekivad andmebaasi salvestamisega.

1. Kui ei kasutata FILESTREAMi andmebaasi salvestamisel on objekti maksimaalne suurus 2GB. See tähendab, et kui tekib vajadus salvestada suuremaid faile siis see tekitab probleeme.

2. Andmebaasi suurus kasvab kiiresti. Üks pilt tähendab tihtipeale tuhandeid ridu tekstiandmeid. Igasugused andmete varukoopiate suurused kasvavad kiiresti ja nende tegemine on aeglasem.

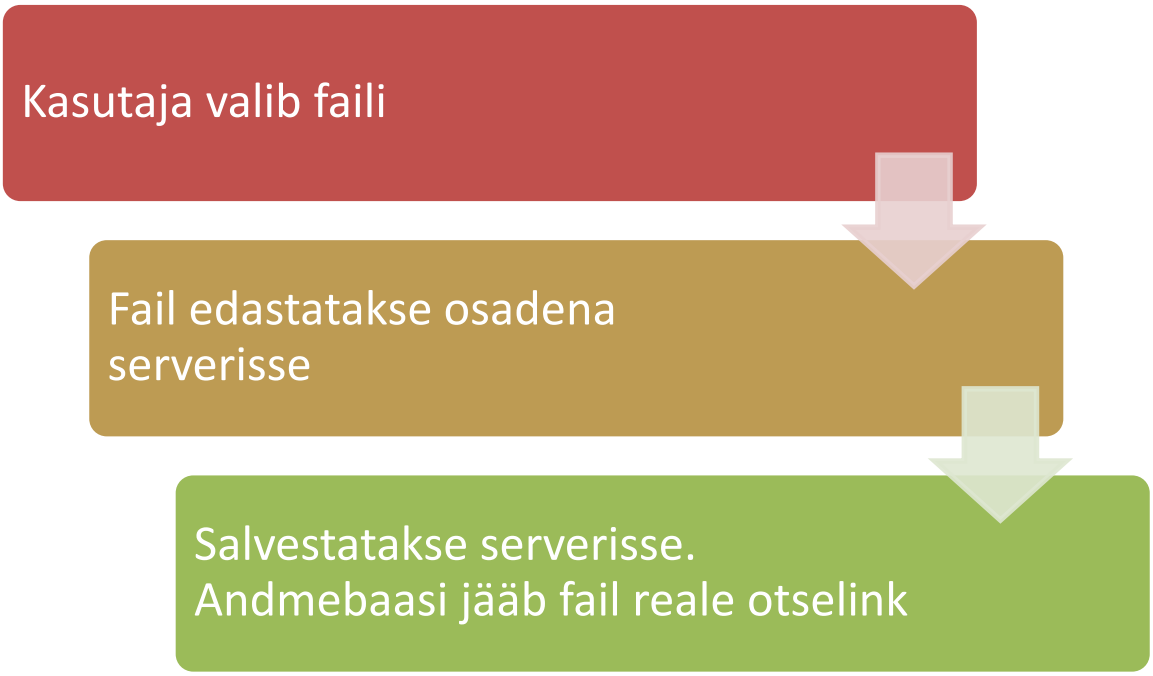
3. Andmete teisaldamine on keerukam olenevalt salvestatud failiformaadist. Kui failid on salvestatud FILESTREAM objektina ja nüüd on vaja salvestata serverisse, mis pole SQL baasil.

4. API koodiga suhtlemine võib erinevate süsteemide vahel tekitada probleeme.

5. Pole võimalust kasutada pilveteenuseid ja sellest tulenevaid eeliseid.

Serverisse salvestamine lubab hoida ka mitut failiserverit muutes andmete skaleerimise kergemaks. API mõttes lihtsustab see suhtlemist teiste teenustega. Piisab vaid lingi andmisest ja iga teenusepakkuja saab ise valida, millist protokolli kasutada ja kuidas andmeid küsida.

Tabel 10: Faili salvestamine serverisse



6.3 Valitud lahendus

Pärast plusside ja miinuste kaalumist jäime kindlalt serverisse salvestamise juurde. Minu jaoks oli kõige tähtsam sealjuures skaleeritav lahendus ja võimalus kasutada pilveservereid. Kuna plaan on ka antud projekti pikemalt laiendada leidsime, et sellisel viisil on teistel meile informatsiooni kirjutada. Esialgne soov on salvestada failid meie serverisse, kuid juba praegu vaatame laiemat võimalust salvestada failid Amazoni serverisse. See vähendaks vastutust ja lubaks kasutada ka juba tuntud APIt. Seadsime pildi maksimaalseks suuruseks 10mb. Soovime lubada korraga mitme faili lisamist ja seetõttu võib tekkida probleeme kui need on suuremad kui 10mb. Valitud protokolliks on PUT ja eeldab parameetritena sessiooni tokenit ja linki failile.

7. Arhitektuur ja tehnoloogia

3. peatükis sai lühidalt kirjeldatud serverit ja andmebaasi, kuhu rakendus loodud. Allolevas peatükis kirjeldan lähemalt API kutsete realiseerimist

API realiseerimiseks sai valitud PHP programmeerimiskeel ja selle mikroraamistik Flightphp. Arutuse all oli ka slim raamistik, mis jäi välja peale väikest eksperimenteerimist. Flighti boonused :

1. Lihtne alustada
2. Toetab täiesti JSON kutseid / vastuseid
3. Toetab vahemälu kasutamist

```php
<?php
require 'flight/Flight.php';

Flight::route('/', function(){
    echo 'hello world!';
});

Flight::start();
?>
```

Joonis 6: FlightPHP „Hello World“ näide [10]

FlightPHP kasutab staatilisi meetodeid, mis tähendab, et klasse pole vaja konkretiseerida. Lisaboonusena on olemas võimaluse filtreerida enne ja pärast iga kutset. See tähendab, et igale kutsele on kerge lisada eelnev kontroll tokeni jaoks.

7.1 Implementatsioon

Eelnevalt sai kirjeldatud kutseid ja nende loomispõhimõtteid. Siinkohal tuleb kirjeldus nende realiseerimisest programmis. Realiseeritud osa koodist on kättesaadaval lingilt: https://drive.google.com/file/d/0B7LOw1SaPFGFLXdyZ3VkcEw1R0E/view?usp=sharing

7.1.1 Suunamine

Vaikimisi kontrollib FlightPHP suunamises iga meetodi vastu. API jaoks on tihtipeale aga tähtis just see meetod, millega kutset tehakse. Samas on võimalus defineerida kutses ka meetod mille vastu kontrollida. Selle jaoks tuleb peale Flight::route kutset ette anda ka parameeter.

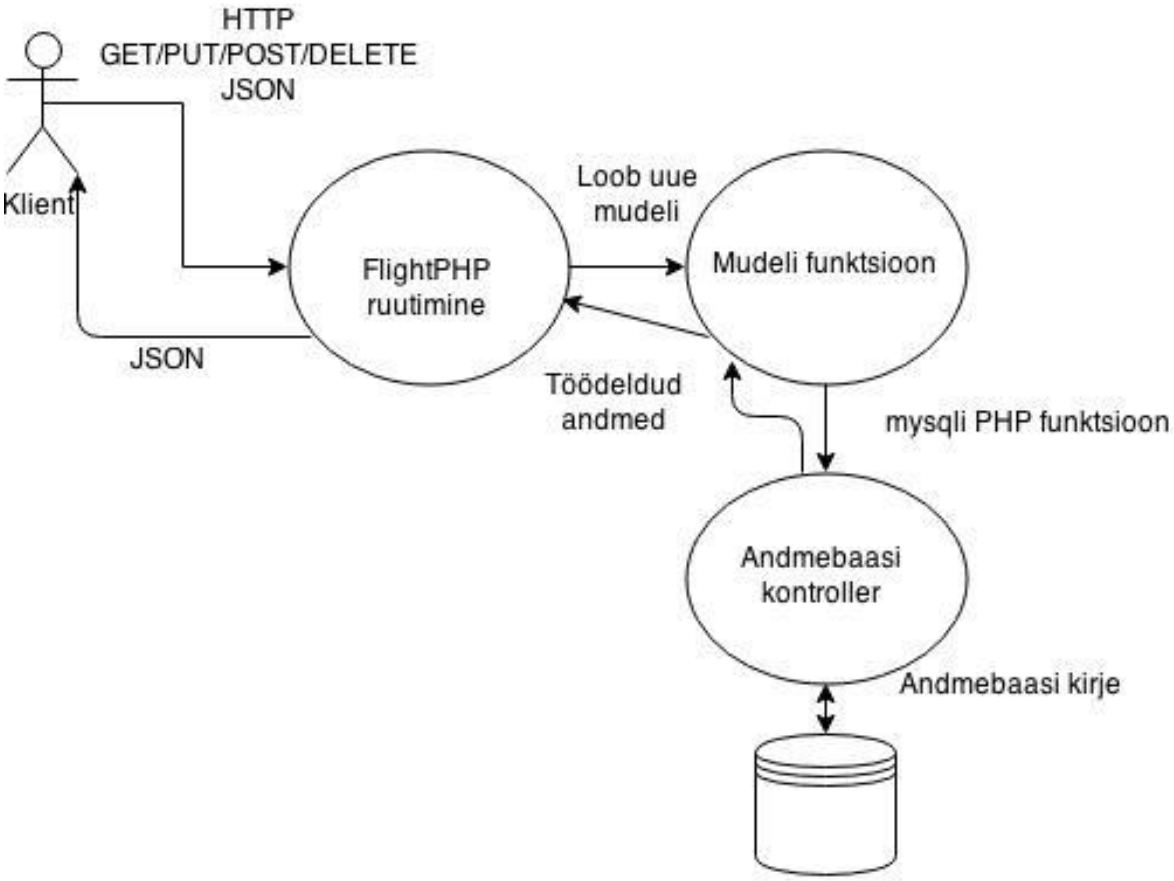
Tabel 11: Meetodid kutsel

Meetod	Kutse	Seletus

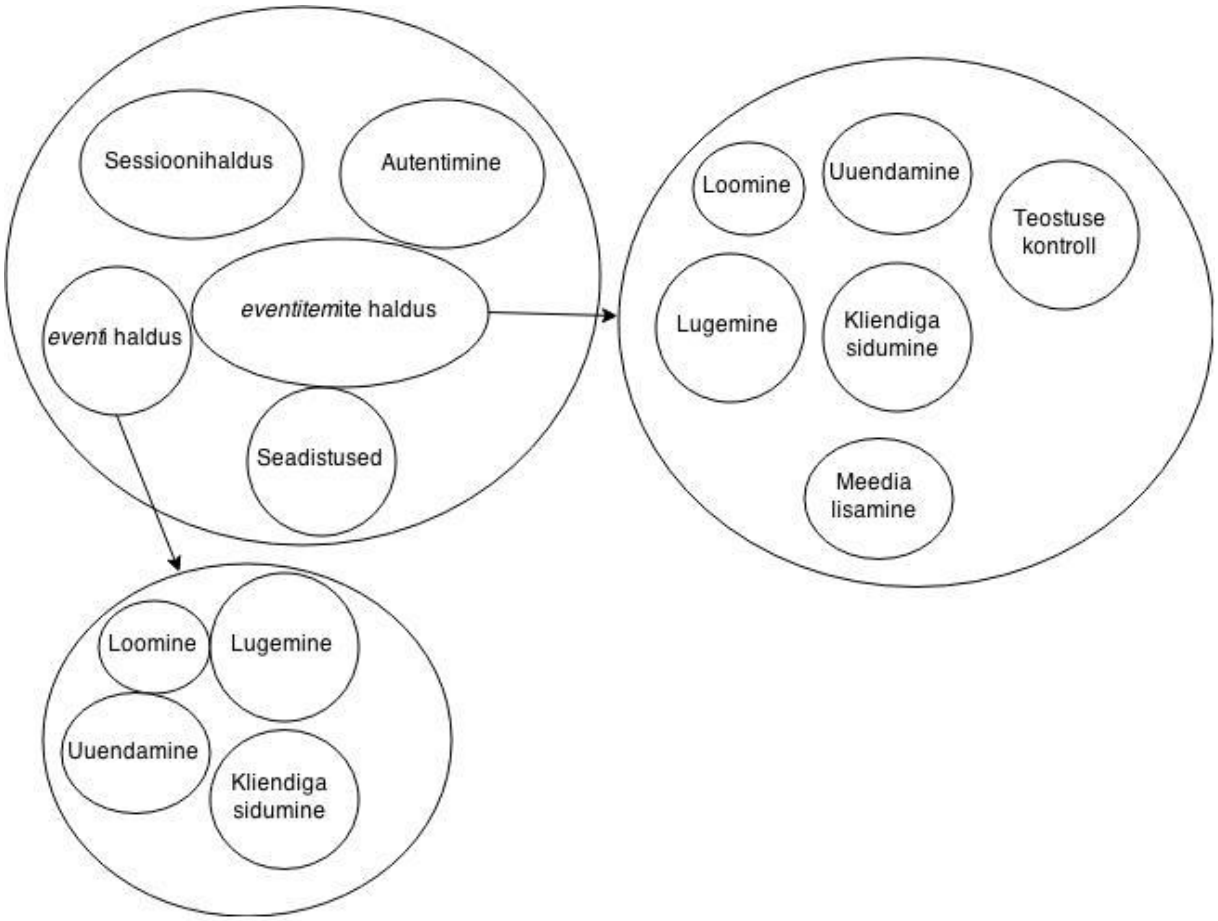
GET	Flight::route('GET /event/@id', function($id){
	$event = new Event();
	print $event->getEvent($id);
	});
		Tagastab JSONina *event*i parameetrid, mille ID ette anti

PUT	Flight::route('PUT /event/@id', function($id, &$params){ $event = new Event(); print $event->update ($id, $params); });	Muudab etteantud parameetreid *event*il

Kuigi kutse on sama on erinevatel meetoditel ka teistsugused funktsioonid.



Joonis 7: Kutse tegemine



Joonis 8: Mudeli funktsioonid

7.1.1.1 Kontrollimised suunamisel

FlightPHP raamistikul on lubatud ka regulaaravaldiste kontroll. See tähendab, et kutsel saab kontrollida, et seda korrektselt välja kutsutakse. Id peab alati olema täisarv ning seda kontrolli saab rakendada nii:

Flight::route('/event/@id:[0-9]'), mis tähedab et id peab sisaldama ainult numbreid 0-9. Võimalik on ära määrata ka pikkus lisades {6}, mis tähendab, et peab sisaldama kuute numbrit.

7.1.2 Laiendamine

FlightPHP on ehitatud laiendatavaks. See tähendab, et selles on võimalik kaardistada oma meetodeid ja registreerida enda klasse. Paindlikkus teeb seda raamistiku tulevikukindlamaks ka meie API suhtes. Näitena valideerimise kaardistamine:

```
Flight::map('validate', function($token){
        $validate = new Token();
       print $validate->validateToken($token);
```

```
});
Flight::validate('a003bc7ad1d2bffca2022af8ea8c04bc')
```

Klasside registreerimist Flightis praeguses skoobis ei kasutanud kuna lõime enda andmeklassid paremaks andmebaasi päringute tegemiseks ja filtreerimiseks. Tulevikus võib see aga ärivajaduste keerukusest sõltuvalt tähtsamaks saada.

```
Flight::register('event', 'Event');

$user = Flight::user();
```

7.1.3 Filtreerimine

FlightPHP teeb mugavaks meetodite filtreerimine enne ja pärast kutset. Selle jaoks on vaja lisada meetod „before“.

```
Flight::before('event', function(&$params, &$output){
    return Flight::validate(params['token']) ?
Flight::validate(params['token']) : Flight::validationException();
});
```

Eelnev näide kontrollib enne iga *event* kutset kas sellele on lisatud token ja valideerib. Negatiivse vastuse korral saadakse valideerimise erand ja ei minda kutsega edasi.

Olemas on ka järgnev kutse, mida kasutame tokeni uuendamiseks peale korrektset kutset. Seda saab välja kutsuda kasutades Flight::after käsku.

7.1.4 Andmebaasiga suhtlus

API jaoks on ehitatud andmebaasi suhtluse lihtsustamiseks üks kontroller. Selle eesmärk on kergem ühendamine andmebaasiga ja päringute realiseerimine. Selle jaoks on kontrolleris funktsioonid, mis lahendavad andmebaasis SELECT, UPDATE ja INSERT päringud. Neid funktsioone kasutatakse mudelis.

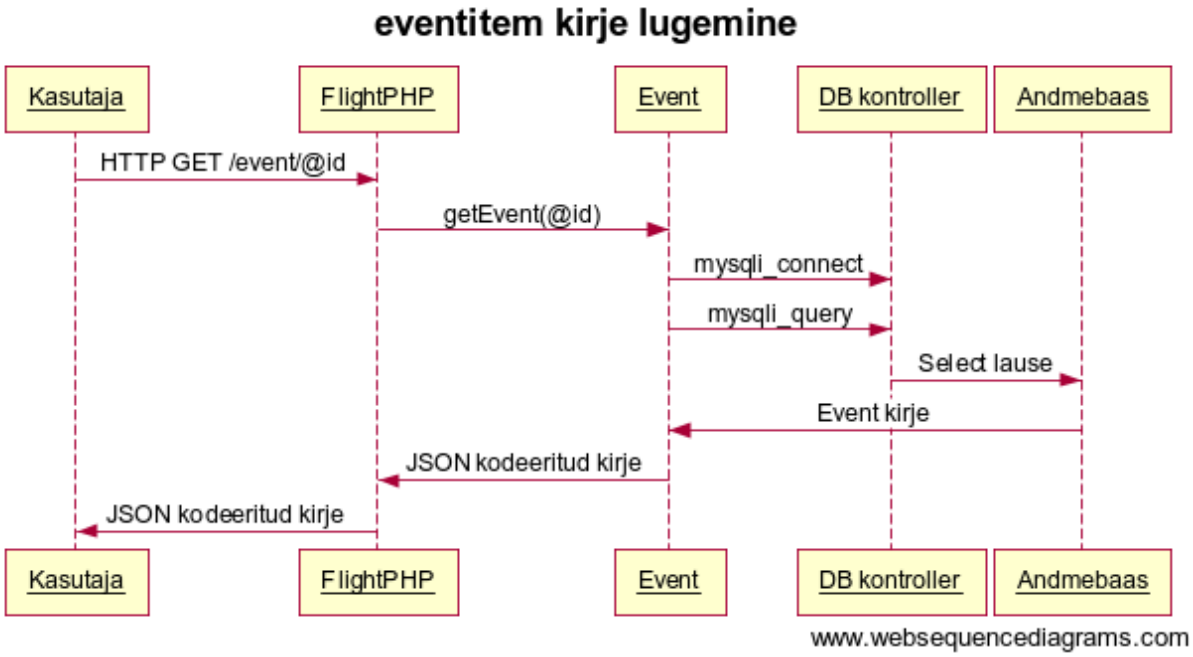
Kontroller realiseerib andmebaasi ühenduse funktsiooniga connect(), mis kasutab PHP sisseehitatud mysqli meetodit andmebaasi ühendamiseks. Parameetrid (server, kasutajanimi, salasõna ja andmebaasi nimi) on määratud konfiguratsioonifailis.

SELECT on realiseeritud läbi select() funktsiooni, mis annab kontrollerile ette MySQL lause, mis tuleks realiseerida.

UPDATE ja INSERT on realiseeritud läbi query() funktsiooni, mis samuti annab kontrollerile ette lause, mis tuleks realiseerida. Ainult seekord on resultaat kas edukas muutmine või erand.

Objektid edastatakse JSONina kasutatades PHP json_encode funktsiooni.

Kui algselt oli soov kasutada RedBeanPHP ORMi, selle suhtelise lihtsuse tõttu siis andmebaasi skeem rikkus selle plaani. Edasi sai vaadatud veidi keerulisemaid ORMe (Doctrine, Yii) kuid nende implementatsioon on küllaltki keeruline väiksemate realisatsioonide jaoks ning lõpuks jäin PHP enda andmebaasiga suhtlemise funktsioonide juurde.



Joonis 9: *eventitem* kirje lugemine

7.1.5 Andmeklassid

Antud API andmeklass on väga lihtne ja koosneb neljast objektist: Event, EventItem, Client ja Settings. Igas andmeklassis on meetod uue objekti loomiseks, lugemiseks ja uuendamiseks. Veel on realiseeritud vastavalt vajadusele eraldiseisvad funktsioonid äriloogika realiseerimiseks. Näiteks *event*i sidumine kliendiga, kus kirjeid tuleb nii uuendada kui ka luua.

Andmete lugemisel tagastatakse JSON tüüpi objekt. Andmete muutmisel või loomisel tagastatakse vastavalt kas teade edukalt lõppenud protsessi kohta või siis veateade.

Settings andmeklassis hoitakse funktsioone, mis määravad ära näiteks rakenduse seadistused ja väljanägemise.

8. Kokkuvõte

Lõputöö eesmärk oli realiseerida API ühele mobiilirakendusele ja uurida erinevaid võimalusi selle tegemiseks. Edasine eesmärk oli leida võimalusi teha loodav API kasutatavaks ka arendajatele, kes soovivad oma rakendustes kasutata platvormi AddGoals andmeid.

Olulisemaks tulemuseks oligi APIle vastavate funktsionaalsuste loomine. Edasiseks tuli ka järeldus, et mitte alati ei saa kasutada kõiki RESTful API põhimõtteid ning tihtipeale tuleb lähtuda vastavast vajadusest.

Lõputööle seatud eesmärgid said täidetud kuid kindlasti on veel võimalus teha APIt paremaks teistele kasutajatele. Koos rakendustega areneb ka API.

Summary

The aim of the work was to create a functional API that can be used by the mobile application and find different ways of doing so. Additional goal was to make that API useable by other developers who wish to add certain functionality to their application.

Most important result was creating the actual API for the application. In addition I realized that RESTful API principles are not definitive guidelines but more as ways to accomplish means. Often I had to think more of the functionality than the realization of said functionality.

I believe the goals that I set for the work got achieved. There are definitely many ways to better the API and additionally the API will keep growing with added functionality to the main application

Kasutatud kirjandus

[1] D. K. C. J. a. J. Higginbotham, „A Practical Approach to API Design,“ %1 *A Practical Approach to API Design*.

[2] J. M. Scott Gowell, „Professional Mobile Application Development,“ %1 *Professional Mobile Application Development* .

[3] „Apple Support,“ Apple, [Võrgumaterjal]. Available: https://support.apple.com/en-gb/HT202880.

[4] „EMT koduleht,“ [Võrgumaterjal]. Available: https://www.emt.ee.

[5] „Tartu Ülikooli arvutiteaduse instituut,“ [Võrgumaterjal]. Available: https://courses.cs.ut.ee/2013/infoturve/fall/Main/Autentimine.

[6] „Technet,“ [Võrgumaterjal]. Available: https://technet.microsoft.com/en-us/library/bb933993%28v=sql.105%29.aspx.

[7] M. L. Mike McBride, „Failide seosed,“ [Võrgumaterjal]. Available: https://docs.kde.org/stable/et/kde-runtime/kcontrol/filetypes/filetypes.pdf.

[8] „Statista,“ [Võrgumaterjal]. Available: http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/.

[9] P. Dave, „Pinal Dave (http://blog.SQLAuthority.com),“ [Võrgumaterjal]. Available: http://blog.sqlauthority.com/2007/12/09/sql-server-acid-atomicity-consistency-isolation-durability/.

[10] „Freshwebdev,“ [Võrgumaterjal]. Available: http://freshwebdev.com/best-restful-micro-php-framework-code-example-tutorial.html.

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

[12] „http://et.wikipedia.org/wiki/Rakendusliides,“ [Võrgumaterjal].

[13] „PHP-guru,“ [Võrgumaterjal]. Available: http://www.php-guru.in/2013/upload-files-using-php-curl/.