TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

IAG40LT

Siim Talts 112354

ARVUTITEHNIKA INSTITUUDI RAAMATUKOGU VEEBITEENUSE PLATVORMI ARENDUS JA ANDROID MOBIILIRAKENDUS

Bakalaureusetöö

Tarmo Robal
Phd
Teadur

Tallinn 2016

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Siim Talts

13.12.2015

Annotatsioon

Antud lõputöö raames on loodud Arvutitehnika instituudi tarbeks kirjanduse haldamise süsteem, mis koosneb veebiteenusest ja Android rakendusest. Töös antakse ülevaade veebiteenuste ja Android rakenduste disainimisel vajalikest aspektidest. Töös valminud veebiteenuse valmistamisel on jälgitud REST arhitektuurile omaseid tunnusjooni. Samuti on teenus loodud olema kasutatav ka muude rakenduste tarbeks peale raamatukogu platvormi Android mobiilirakenduse. Loodud Android rakendus on instituudi raamatutebaasi loomiseks vajalik tööriist, millega on võimalik kirjandust lisada ja hallata. Samuti loob rakendus tervikliku ülevaate olemasolevatest raamatutest ning võimaldab korraldada ka nende välja laenutamist. Lisaks ei vaja kasutajapõhine rakendus uue konto loomist, vaid on võimalik kasutada ITA siseveebi kasutajakontot. Siiski vajaksid nii veebiteenus kui ka rakendus paremate tulemuste saavutamiseks mõningaid täiendusi, et parandada nende funktsionaalsust ja kasutusmugavust.

Töö sisaldab endas loodud veebiteenuse arhitektuuri ja kasutusjuhendit ning Android rakenduse kasutusjuhtude kirjeldusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 7 peatükki, 17 joonist.

Abstract

Development of a web service platform and Android application for the library of Department of Computer Engineering

Many today existing mobile applications are created to assist their users by simplifying and optimizing everyday activities. The time when a smart phone was mainly used for calls, messages and for gaming at free time is over. Our phones have become a tool like computers, without which some operations would be incredibly complex to complete. However, for example Android applications capabilities are quite restricted where the goal is to create a Web-based and real-time solution. To overcome that obstacle, it is necessary to create a symbiosis between the web service in the web server and the application inside the smart phone. For example, phone application for warehouse management systems could need support from a web service to complete almost every single task.

The outcome of this thesis is a library management system for the Department of Computer Engineering, which consists of a web service and an Android application. The thesis will give an overview of the necessary aspects which should be followed while creating web services and Android applications. REST architectural requirements have been flowed while creating the web service for the library platform. Also this service is meant to be used by other applications besides the Android application, which was created for this thesis, which require changing information with the database in the Department of Computer Engineering. The Android application which was created is a tool for creating a central database of existing books. It also creates a unified perspective of existing books for all the users and manages lending literature out. In addition, the application does not require users for creating new accounts, but reuses the existing user accounts of ITA intranet. However, both the web service and Android application could need some extra developments for improving their functionality and ease of use in order to achieve better results.

The thesis includes a manual for using the web service, its architectural overview and descriptions of use cases for the Android application.

The thesis is in Estonian and contains 47 pages of text, 7 chapters, 17 figures.

Lühendite ja mõistete sõnastik

Apache	Veebiserveri vabavara
API	*Application Programming Interface,* rakendusliides, reeglistikud, protokollid ja tööriistad olevasoleva tarkvaraga suhtlemiseks
APK	*Android application package,* pakettfaili formaat, mida kasutab Android operatsioonisüsteem programmide levitamiseks ja paigalamiseks
CRM	*Customer relationship management,* kliendisuhtluse haldur
DoS	*Denial-of-Service Attack,* teenusetõkestamise rünne, arvutisüsteemi või võrgu vastu suunatud rünnak, mis koormab võrku suure tarbetu liiklusega
Cron job	Veebiserveris töötava programmi Cron, mis on ajapõhiste tööde haldur, üks töö, mis käivitatakse määratletud perioodi tagant
CRUD	*Creat, read, update, delete,* püsimäluga teostatavad baasoperatsioonid
HTTP	*Hypertext Transfer Protocol,* Hüperteksti edastusprotokoll, protokoll andmete edastamiseks arvutivõrkudes
ID	Süsteemne identifikaator, mida kasutab programm või selle osa
IP	Internetiaadress võrku ühendatud arvutil või muul seadmel
ISBN	*International Standard Book Number,* raamatu rahvusvaheline standardnumber, 13-kohaline, unikaalne numbrikombinatsioon
ITA	IT Akadeemia, koostööprogramm ja kaubamärk, mis aitab tõsta IKT kõrghariduse kvaliteeti ning turustada IKT alast kõrgharidust
JMS	*Java Message Service,* Java sõnumite saatmise teenus, Java platvormil töötav sõnumite saatmise protokoll

JSON	*JavaScript Object Notation,* avatud standardiga andmevahetusformaat, kergkaaluline alternatiiv XML-le
MSDN	*Microsoft Developer Network,* osa firmast Micorsoft, mis vastutab suhtlemise eest tarkvaraarendajatega ja testijatega
mySQL	Populaarne avatud lähtekoodiga relatsioonilise andmebaasi haldamise süsteem
PHP	*Hypertext Preprocessor,* platvormist sõltumatu programmeerimiskeel, mis töötab veebiserveris
REST	*Representational State Transfer,* tarkvara arhitektuuri stiil, mis koosneb juhenditest ja reeglitest, et koostada skaleeritav veebiteenus
ROA	*Resource Oriented Arhitecture,* ressursile orienteeritud arhitektuur, tarkvara arendamise stiil, mis põhineb ressurssidel ja REST liidestel
RPC	*Remote Procedure Call,* kaugprotseduur, protseduuri keskne kommunikatsioon, mis võimaldab eemal olevas arvutivõrgus käivitada teise protseduuri
SDK	*Software Development Kit,* arendustarkvara, mis võimaldab programmerijatel luua rakendusi konkreetsele platvormile
SMTP	*Simple mail transfer protocol,* lihtne meiliedastusprotokoll, kasutatakse e-kirjade edastamiseks üle arvutivõrkude
SOA	*Service-oriented architectures,* teenus-orienteeritud arhitektuur, arhitektuuriline muster mille komponendid pakuvad teenuseid teistele komponentidele üle arvutivõrkude
SOAP	*Simple Object Access Protocol,* lihtne objektpöördusprotokoll, arvutivõrkudes kasutatav protokoll millega veebiteenused vahetavad struktuurseid andmeid

SQLite	Relatsiooniline andmebaasi haldamise süsteem
TCP	*Transmission Control Protocol,* edastusohje protokoll, levinuim arvutivõrkude transpordikihi võrguprotokoll
UDDI	*Universal Description Discovery and Integration protocol,* platvormist sõltumatu XML-l baseeruv register kuhu erinevad ettevõtted ülemaailma saavad oma veebiteenuseid avalikustada
URL	*Uniform Resource Locator,* internetiaadress, mis vastab igale dokumendile või ressursile internetis
W3C	*World Wide Web Consortium,* rahvusvaheline organisatsioon, mille eesmärgiks on arendada ja standardiseerida veebi
WSDL	*Web Services Description Language,* XML-le baseeruv keel, mis kirjeldab veebiteenuse funktsionaalsust
XML	*Extensible Markup Language,* W3C poolt väljatöödatud üldotstarbeline märgistuskeel, mille eesmärgiks on info jagamine erinevate infosüsteemide vahel

Sisukord

Jooniste nimekiri

1. Sissejuhatus

Paljud täna olemas olevad mobiilirakendused on loodud kasutajaid abistama, muutes igapäevased tegevused lihtsamaks ja optimaalsemaks. Aeg, mil nutitelefon oli vaid seade millega sai helistada, sõnumeid saata ning vabal ajal mängida, on läbi. Telefon on muutunud samasuguseks tööriistaks nagu arvuti, milleta teatud tegevusi on väga keeruline teha. Siiski on näiteks Android rakenduste võimekus suhteliselt piiratud, kui on eesmärgiks luua veebipõhine ning reaalajas uuenev lahendus. Selle takistuse ületamiseks on vaja luua sümbioos veebiserveris olevast veebiteenusest ja telefonis asuvast rakendusest. Eriti tugevalt vajavad veebiteenuse tuge näiteks erinevad haldussüsteemid.

Käesoleva bakalaureusetöö peamiseks ülesandeks oli luua Tallinna Tehnikaülikooli Arvutitehnika instituudile üha suureneva õppekirjanduse baasi paremaks haldamiseks raamatukogu veebiteenuse ja Android rakenduse prototüüp. Samuti on siht loodav veebiteenus disainida võimalikult dünaamiline, et ka muud süsteemid peale raamatukogu Android rakenduse saaksid seda muudel eesmärkidel kasutada. Android rakenduse eesmärgiks on luua mugav tööriist raamatutebaasi loomiseks ja haldamiseks, mis tekitaks tervele instituudile selge ülevaate olemasolevast kirjandusest, ning oleks seotud instituudi portaaliga mis välistaks täiendavate kasutajakontode loomise.

Bakalaureusetöö teises ja kolmandas peatükis antakse ülevaade veebiteenuste arendamise aspektidest ning tutvustatakse Android platvormile rakenduste loomist. Arvutitehnika instituudi raamatukogu jaoks loodud veebiteenuse arhitektuuri ja kasutusjuhendit kirjeldatakse neljandas peatükis ning Android rakenduse tööpõhimõtteid ja kasutusjuhtusid viiendas peatükis. Töö kuues peatükk analüüsib loodud veebiteenust ja Android rakendust lähtudes esimesest kasutamise tagasisidest ning kirjeldab süsteemi nõrku kohti.

2. Veebiteenused

2.1. Veebiteenuse definitsioon

Veebiteenused on W3C (World Wide Web Consortium) poolt defineeritud kui tarkvara süsteem, mille eesmärgiks on toetada rakenduste vahelist suhtlust üle arvutivõrgu [1]. Veebiteenus teenindab ülejäänud rakendusi. Enamus veebiteenuseid on võimalik kasutada ainult väga piiratud keskkonnas, ehk siis ühe rakenduse või rakenduste raames, kuid on palju teenuseid mis on avatud kõikidele kasutamiseks.

Veebiteenused on reeglina erinevatest programmeerimiskeeltest ja platvormidest sõltumatud ning opereerivad HTTP (Hypertext Transfer Protocol - hüperteksti edastusprotokoll) protokolli peal. Tänapäeval on enamus veebiteenuseid standardiseeritud. W3C tunnustab kahte sorti veebiteenuseid – REST (Representational State Transfer - tarkvara arhitektuuri stiil) nõuetele vastavad ja isevaldsed. Veebiteenus nõuab sisendiks kindlaks pandud skeemiga päringut ning tagastab struktureeritul kujul vastuse. Levinumad formaadid on XML (Extensible Markup Language - üldotstarbeline märgistuskeel) ja JSON (JavaScript Object Notation - andmevahetusformaat). Loodava tarkvara kasutajaliides võidakse ehitada veebiteenuse peale, ehk see võib kasutada veebiteenust, aga veebiteenus ise ei sisalda graafilist kasutajaliidest.

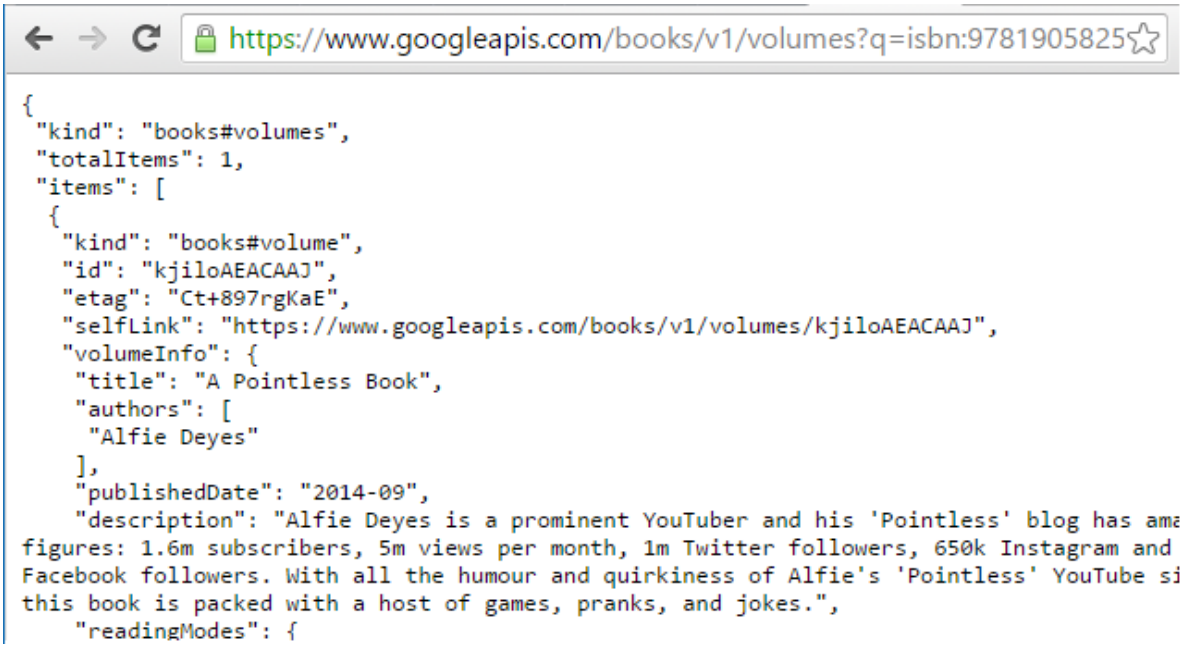
Kuna veebiteenused töötavad HTTP protokolli peal, on vaja kasutada teatavaid käsklusi, et opereerimine HTTP peal oleks võimalik. Selleks kasutatakse HTTP meetodeid, mis on HTTP versiooni 1.1 sisse ehitatud. Kokku on kaheksa meetodit ning nende nimed on tõusutundlikud:

- GET, seda meetodit kasutatakse, et tagastada informatsiooni mingist serverist. GET päringut kasutades peaks ainult tagastama andmeid, mitte neid kirjutama kuhugi
- HEAD, tagastab näiteks GET päringu staatuse ja päise osa
- POST, kasutatakse andmete saatmiseks üle võrgu näiteks HTML vormide kaudu
- PUT, kasutatakse andmete saatmiseks, asendab kõik sama kirjega seotud andmed
- DELETE, kasutatakse andmete eemaldamiseks
- CONNECT, kasutatakse ühenduse loomiseks mingi serveriga

- OPTIONS, tagastab HTTP meetodid mida server toetab defineeritud URL-i (Uniform Resource Locator - internetiaadress) puhul
- TRACE, teostab ühenduse kontrolli

Veebiteenuste puhul on kõige rohkem kasutatavad meetodid GET, POST, PUT ja DELETE.

Veebiteenuseid on loodud väga erinevate valdkondade jaoks. On teenuseid mis teenindavad suuri ja võimsaid CRM-e (Customer relationship management - kliendisuhtluse haldur) ning on ka väga väikseid ja rangelt piiritletud veebiteenuseid nagu näiteks Google Books API (Joonis 1), mis tagastab raamatutega seotud informatsiooni.



Joonis 1. Google Books API veebiteenuse näidispäring

2.2. Veebiteenuste eesmärk ja vajalikkus

Veebiteenuse eesmärk on luua kindla funktsionaalsusega HTTP protokolli peal töötav rakendus, mis oleks kättesaadav ja kasutatav üle võrgu teistele tarkvaradele. Veebiteenuse võlu peitub selles, et peale seda, kui teenus on võrgus kättesaadav, ei ole tähtis millist programmeerimiskeelt, tarkvararaamistiku, platvormi või riistvara kasutatakse, et teenusega suhelda.

Realiseerides tarkvaralise rakenduse veebiteenusena, annab see võimaluse kõigile, kes kasutavad HTTP protokolli suhtlemiseks ning omavad teenusele ligipääsu, saada kasu rakendusest. Samuti mõjub teenuse olemasolu ka teenuse haldajatele kasulikult, sest üle arvutivõrgu teenust kasutades on kasutajate hulk oluliselt suurem.

Paljudel veebiteenusena töötavatel rakendustel on realiseeritud ka kasutajaliides, millele pääseb ligi näiteks internetibrauserist ning on võimalik ilma programmeerimiseta teenust kasutada. Ka peatükis 2.1 viidatud Google Books teenust on võimalik kasutada brauserist. Siin tasub tähele panna, et teenus ise ei sisalda endas kasutajaliidest. Antud näite puhul kasutab brauseris töötav rakendus veebiteenust, et kasutajaliidesel kuvada tulemusi. Sisuliselt tähendab see seda, et kasutades veebiteenust on võimalik igaühel luua sarnase funktsionaalsusega rakendus. Selliseid veebiteenuseid, mis piirnevad ainult mingi teatud rakenduse funktsionaalsuse kasutamisega nimetatakse ka rakendusliidesteks ehk API-deks.

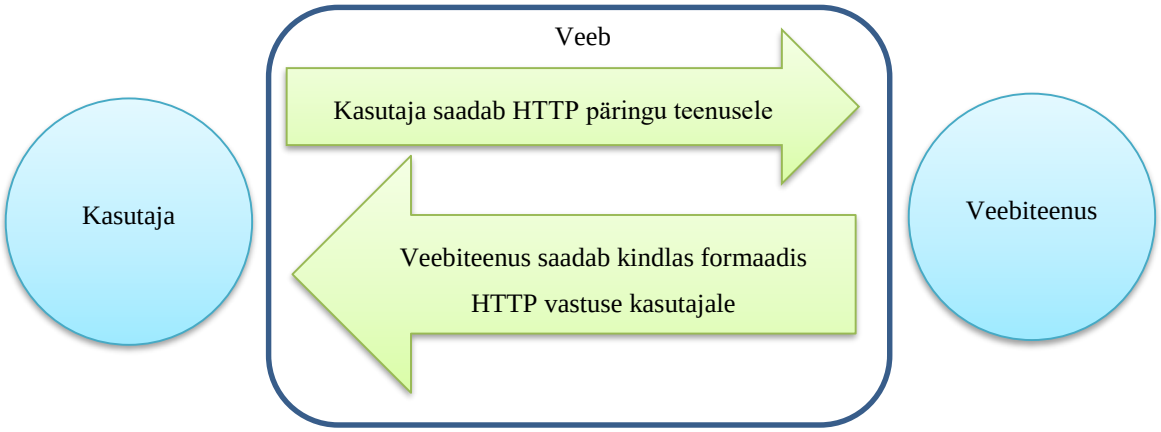
API (Application Programming Interface - rakendusliides) on töötava rakendusega suhtlemiseks loodud liides, mis võimaldab kolmandatel osapooltel lugeda, kirjutada ja uuendada rakenduses olevaid andmeid, kui see on kasutajale võimaldatud. Samuti teeb API olemasolu võimalikuks rakendusele lisafunktsionaalsuse loomise kolmandatel osapooltel. Rakendusele API funktsionaalsuse lisamine teeb võimalikuks ka tervest süsteemist vaid teatud komponendi kasutamise, kuna korrektsete päringutega tagastab kapseldatud rakendus vaid need andmed mida küsiti. Tähelepanu tasub pöörata asjaolule, et veebiteenus kasutab opereerimiseks pea alati HTTP protokolli, siis API puhul see nii ei ole.

2.3. Veebiteenuste tüübid

Iga veebiteenuse kasutamise initsialiseerib veebiteenuse kasutaja (Joonis 2). Veebiteenuste tüüpideks [2] võib lugeda standardeid mida kasutatakse veebiteenust luues ning teenuse tööpõhimõtteid. Veebiteenust luues tuleks valida selline standard mis on kõige sobilikum teenuse iseloomuga. Antud töös on kasutatud portaali ProgrammableWeb[1] andmeid, et võrrelda erinevate protokollide populaarsust veebiteenustes, mille andmetel on kõige populaarsemad veebiteenuste protokollid 2015 aasta maikuu seisuga SOAP (Simple Object

[1] http://www.programmableweb.com/

Access Protocol - lihtne objektpöördusprotokoll) ja REST (REST 60% ja SOAP 23%), kuid võrreldes 2012. aasta andmetega on REST osakaal tasapisi hakanud siiski langema. ProgrammableWeb on internetis asuv andmebaas avalikest API-dest. Selle andmed kogunevad registreerinud kasutajatelt saadud info põhjal ning ei kirjelda olukorda täielikult korrektselt, kuid annab hea ülevaate sellegipoolest erinevatest trendidest.



Joonis 2. Lihtsustatud veebiteenuse kasutamise mudel

Tegelikkuses ei ole alati võrdlus SOAP ja REST vahel õigustatud, sest SOAP on protokoll ning REST on arhitektuuriline stiil veebiteenuste kompositsiooniks. Siiski on need kaks ühed levinumad lähenemised veebiteenuste ehitamisel.

2.3.1 SOAP

SOAP on osa teenus-orienteeritud arhitektuuridest ning on ainult üks paljudest SOA (Service-oriented architectures - teenus-orienteeritud arhitektuur) tehnoloogiatest. SOA on kogum põhimõtteid ja meetodeid veebiteenuste disainimiseks ja arendamiseks. SOA üks ideedest on luua modulaarsed veebiteenused, reklaamida neid tsentraalses registris ning lasta kasutajal valida endale sobiv. Veebiteenuste kirjeldamiseks on loodud XML formaadis keel WSDL (Web Services Description Language - veebiteenuse funktsionaalsuse kirjeldus). Samuti on võimalik teenuse WSDL lisada UDDI (Universal Description Discovery and Integration protocol – veebiteenuste register) abil teenuste registrisse, et kolmandad osapooled seda veebiteenust lihtsasti kasutada saaksid.

SOAP on protokoll mida saab kasutada HTTP protokollil, et täide viia RPC (Remote Procedure Call, kaugprotseduur) stiilis kommunikatsiooni kliendi ja serveri vahel. Oma lihtsamail kujul on SOAP meetod vahetada sõnumeid üle arvutivõrgu ning seega on see sõltumatu transpordikihist. SOAP-i saab kasutada ka SMTP (Simple mail transfer protocol - lihtne meiliedastusprotokoll), TCP (Transmission Control Protocol - edastusohje protokoll) ja JMS (Java Message Service - Java sõnumite saatmise teenus) protokollidega. RPC põhimõtetel töötab SOAP järgmiselt: klient kirjeldab sõnumiga oma tegevuse, saadab selle üle HTTP serverile ning seal transleeritakse sõnum ja täidetakse tegevus.

Vaikimisi on SOAP-i sõnumi formaadiks XML ning kui transpordi kihiks on HTTP, siis kasutatakse POST meetodit, et sõnum serverisse saata. XML dokument sisaldab endas funktsiooni mida soovitakse käivitada ja parameetreid selle jaoks. Server dekodeerib sõnumi ja käivitab funktsiooni ning tagastab tulemi XML formaadis kliendile. SOAP-s kutsutakse selliseid sõnumeid ümbrikuteks. Järgeval näitel (Joonis 3 ja Joonis 4) soovib klient liita numbri stringiga kasutades funktsiooni *function.* Vastuse tagastab funktsioon *functionResponse.*

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding"
 xmlns:SOAP-ENV= "http://schemas.xmlsoap.org/soap/envelope"
 xmlns:SOAP-ENC= "http://schemas.xmlsoap.org/soap/encoding"
 xmlns:xsi= "http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd= "http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
       <examplenamespace:function
       xmlns:examplenamespace="urn:MySoapService">
             <param1 xsi:type="xsd:int">12</param1>
             <param2 xsi:type="xsd:string">-months</param2>
       </examplenamespace:function>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Joonis 3. SOAP ümbriku sisu, mis saadetakse veebiteenusele

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
       <examplenamespace:functionResponse
       xmlns:examplenamespace="urn:MySoapService"
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
             <return xsi:type="xsd:strin">12-months</param1>
       </examplenamespace:functionResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Joonis 4. SOAP formaadis vastus veebiteenusel

Luues veebiteenust, mis kasutab SOAP protokolli on kõige tähtsam standard SOAP ise, XML skeem ja WSDL, mis iseloomustab teenust. WSDL dokument on kindlasti oluline SOAP teenuse arendamisel, kuid moodsaid arendustööriistu kasutades on võimalik WSDL peita ning arendajale graafiliselt veebiteenust kirjeldada.

SOAP loob ühe lisa abstraktsiooni taseme HTTP keskkonda rakendades veel ühe protokolli HTTP peale. Seeläbi käitub SOAP rohkem nagu traditsiooniline RPC. Kuna SOAP-s on realiseeritud funktsioonid, siis on see oma iseloomult väga sarnane traditsiooniliste programmeerimiskeeltega. Seeläbi on SOAP-i kasutamine programmeerijatele üpris mugav.

2.3.2 REST

Kui SOAP oli protokoll, siis REST on üks ROA (Resource Oriented Arhitecture - ressursile orienteeritud arhitektuur) arhitektuurilistest stiilidest. REST liides on seotud ressurssidega vastupidiselt SOAP-le kus seotud olid moodulid ja funktsioonid. Need liidesed on ligipääsetavad ja kasutatavad HTTP meetoditega ning ressursi poole pöördutakse URL-i kaudu. REST liidese võtme karakteristik on rakenduse selgelt määratletav olek kliendi ja serveri vahel.

REST liideses saab kõiki ressursse kuvada URL-ga. Ressurss võib olla staatiline (/news/2015-05-20) või dünaamiline (/news/newest). REST ei kohusta mingit teatud URL

skeemi kasutama, kuid eelistatud on võimalikult inimloetavad variandid. Siiski ei tohiks URL sisalda operatsioone, mida ressursi peal tahetakse pruukida (/add/ või /modify/).

Kõige enam kasutatavad HTTP meetodid REST liideste puhul on GET, POST, PUT ja DELETE. Nende meetoditega saab toime panna kõik CRUD (Cread, read, update, delete) operatsioonid. Meetodeid saab kategoriseerida vastavalt ressurssidele ning REST ei kohusta kõigi meetodite kasutamist.

REST on rangelt olekuta ning rakenduse olek hoiustatakse ainult kliendi poolel. Server hoiustab ainult ressursside olekuid. See tähendab, et iga päring peab sisaldama kogu informatsiooni mis on vajalik päringu läbiviimiseks. Server ei tohiks vajada informatsiooni eelmiste päringute kohta. Iga päring peab olema teistest isoleeritud.

REST puhul on väga oluline ja unikaalne karakteristik see, et REST teenused avalikustavad ressursid teatud hulga HTTP meetoditega teistele kasutamiseks. Kõik on orienteeritud ressurssidele mitte enam funktsioonidele nagu SOAP puhul oli.

Järgnevas näites (Joonis 5 ja Joonis 6) on eelnevalt SOAP puhul läbitehtud näite kordus, kuid seekord kasutades REST tüüpi lähenemist veebiteenusele. Antud näite puhul on kuvatud ka HTTP päis, et veelkord rõhutada, et REST on väga HTTP põhine arhitektuur.

```
> GET /function?param1=12&param2=-months HTTP/1.1
> User-Agent: curl/7.21.4
> Host: localhost: 8080
> Accept: */*
```

Joonis 5. HTTP päring REST arhitektuurilisele veebiteenusele

```
> HTTP/1.1 200 OK
> Transfer-Encoding: chunked
> Date: Wed, 20 May 2015 11.59.42
> Server: localhost
12-months
```

Joonis 6. HTTP vastus REST arhitektuuril olevalt veebiteenuselt

2.4. Veebiteenuste turvalisus

Korrektse ja efektiivse kasutamise korral võib veebiteenusest saada võimas tööriist, mis aitab rakendustel töötada või on abiks mingi olemasoleva tarkvara kasutamisel läbi selle API. Kogu see ressurss, kas valedes kätes või ebakorrektselt kasutades võib põhjustada väga palju probleeme alustades serveri üle koormamisest, valede kirjete lisamisest või hoopis andmete kuritahtliku ära kasutamise. Selle vältimiseks tuleb iga teenuse loomisel lisada teenusesse endasse üks abstraktsiooni tase mis tegeleb turvalisuse tagamisega.

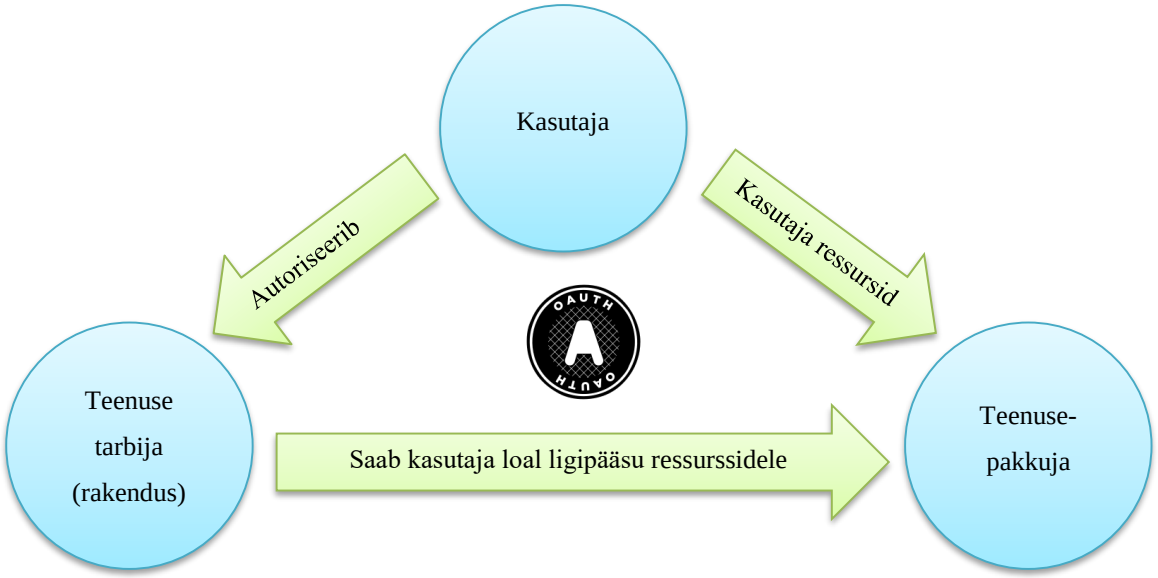
Tarkvarafirma Microsofti[2] MSDN (Microsoft Developer Network) poolt välja antud juhendis „Improving Web Services Security: Scenarios and Implementation Guidance for WCF“ [3] peatükkis „Security Fundamentals for Web Services“, mis käsitleb veebiteenuste turvalisuse küsimust, on loetletud turvalisuse alused:

- **Autentimine.** Protsess mille käigus identifitseeritakse veebiteenuse klient ning määratakse talle mingi unikaalne identifikaator. Autentimist vajavad lõppkasutajad, teised teenused, protsessid ja arvutid.
- **Autoriseerimine.** Protsess mille käigus tehakse kindlaks millistele ressurssidele ja operatsioonidele on autenditud kliendil ligipääs ning mis on tema piirangud. Ressursid võivad olla failid, andmebaasid, tabelite read või tulbad. Operatsioonideks võivad olla näiteks kirjete lisamine või uuendamine.
- **Auditeerimine.** Protsess mille käigus logitakse veebiteenust kasutava kliendi tegevusi. Seda informatsiooni võidakse kuvada nii kliendile ning on ka teenuse haldajatele näiteks teenuse tarbimise maksustamiseks vajalik.
- **Konfidentsiaalsus.** Protsess mille käigu tehakse kindlaks, et kõik andmed jääksid privaatseks ja neid ei saaks kasutada ligipääsu mitte omavad kolmandad osapooled.
- **Puutumatus.** Garantii, et kõik andmed jäävad puutumatuks ning nendes ei toimuks tahtlikuid või mittetahtlikuid muutusi. Eriti oluline on jälgida, et andmete liigutamisel üle võrgu ei tekiks andmete sisus olulisi mittevajalike muudatusi.
- **Kättesaadavus.** Turvalisuse vaatenurgast tähendab kättesaadavus seda, et veebiteenusele ja andmetele oleks autoriseeritud klientidel alati ligipääs olemas.

[2] https://www.microsoft.com

Näiteks DoS (Denial-of-Service Attack - teenusetõkestamise rünne) rünnakute puhul just seda aspekti proovitaksegi häirida, ehk tekitatakse olukord, kus veebiteenuse server on ülekoormatud ning kasutajatel ei ole enam sellele ja seal paiknevatele andmetele ligipääsu.

Autentimine on esimene samm veebiteenusega ühendamisel ning sealne protseduur on vastavalt veebiteenuse iseloomule. Näiteks Google Books API-d[3] on võimalik kasutada ilma autoriseerimata, kuid sellisel juhul on kliendile rakendatud mahupiirangud. Siiski on võimalik taotleda ka API võti või autoriseerida end kasutades oAuth-i. oAuth on autoriseerimise protokoll, mis võimaldab kinnitada ühe rakenduse suhtlemist teisega ilma parooli avalikustamata teisele osapoolele. Selle asemel kasutatakse spetsiaalsete võtmete ja arenduskasutajate süsteemi, mis tagab piiratud ligipääsu ressurssidele ilma liigse kasutaja informatsiooni jagamise (Joonis 7).



Joonis 7. oAuth autoriseerimise protokoll

Autoriseerimise puhul on kõige tähtsam, et teenuse kasutajal ei oleks ligipääsu andmetele ja ressurssidele mis talle ei ole ette nähtud. Selleks tuleb turvalisuse tasemele tekitada autoriseerimise abstraktsioon mis pidevalt kontrolliks teenuse kasutaja õigusi. Kõige tavalisemad õigused on näiteks andmete kirjutamine ja uuendamine ning ka ligipääs teiste kasutajate loodule.

[3] https://developers.google.com/books/

Suurem osa veebiteenuseid omab mingeid piiranguid. Näiteks mahupõhine piirang võib piirata päringute arvu määratletud perioodi vältel. Nende üheks põhjuseks võib olla äriline kuid teiseks on kindlasti teenuse ja serveri stabiilsuse tagamine. Piirangud tagavad selle, et keegi ei hakka tahtmatult või tahtlikult teenust ära kasutama näiteks DoS rünnakuga. Google Books API-l[3] on näiteks mahupiiranguks määratud 1000 päringut päevas (seda on võimalik suurendada) ja 20 päringut sekundis. Iga päring logitakse ning selle põhjal on teenuse kasutajaid võimalik profileerida. Samuti tuleks logida ka ebaõnnestunud päringuid. Antud andmeid saab teenuse haldaja kasutada ka teenuse parendamiseks suurendades näiteks serveri läbilaske võimekust, kui selleks tekib vajadus.

Konfidentsiaalsus ja puutumatus saavutatakse tihtipeale andmete krüpteerimisega. Kindlasti tuleks kasutada erinevate salasõnade, andmebaasi võtmete ja muud tundlikud andmed krüpteerida. Siiski tuleks veebiteenuse loomisel võimalus vältida selliste andmete hoiustamist, mis vajavad krüpteerimist, et hoiduda vigade ja probleemide tekkimisest.

Kättesaadavuse säilitamine on veebiteenuse kõige olulisem funktsionaalne nõue. See saavutatakse kõige eelnevaga, kuid samuti tuleb arvestada lisaks tarkvaralistele nõudmistele ja funktsioonidele ka riistvaralise võimekusega. Töötav teenus peab olema kättesaadav ööpäevaringselt üle terve maailma. Siiski ei tohiks ühtegi teenust saada lõpmatult kasutada ning tuleks implementeerida teenuse kasutamise sessioone, millel on kindel pikkus. Näiteks oAuth võtmete kehtivusaega saab kontrollida teenusepakkuja.

2.5. Veebiteenuse kasutamine Google Books API näitel

Google Books API on veebiteenus, mille abil saab ligipääsu veebi raamatukogule Google Books[4]. See on vaid üks väga mitmetest Google veebiteenustest[5] ning pakub võimalust integreerida Google Books teenust oma rakendusega või leida informatsiooni selle andmebaasiga.

Books API on REST tüüpi veebiteenus ning võimaldatud on 5 operatsiooni:

[4] https://books.google.com/
[5] https://developers.google.com/apis-explorer/

- *List,* HTTP GET meetod, tagastab määratud alamhulga kirjeid
- *Insert,* HTTP POST meetod, lisab uue kirje
- *Get,* HTTP GET meetod, tagastab määratud kirje
- *Update,* HTTP PUT meetod, uuendab määratud kirjet
- *Delete,* HTTP DELETE meetod, kustutab määratud kirje

List ja Get operatsiooni on võimalik piiratud kasutusega tarbida ilma autentimata, kuid ülejäänud operatsioonid nõuavad autentimist. Seda saab teha Google Developers keskkonnas või kasutades oAuth autentimis protokolli. Andmete struktuuri formaadiks on JSON ning Google Books API baas-URL on https://www.googleapis.com/books/v1/.

Päringu, mida saab initsialiseerida URL-ga https://www.googleapis.com/books/v1/volumes ?q=isbn:9781905825905&projection=lite on kitsenduseks määratud ära raamatu ISBN (International Standard Book Number - raamatu rahvusvaheline standardnumber) kood ning parameeter *projections* väärtus on *lite,* mille peale tagastatakse üks kirje lühemas formaadis. Sellise päringu peale saadetakse JSON formaadis vastus (Joonis 8), kus on tähtsamad atribuutid ja nende väärtused olemas. Tehes päringu URL-le mis on vastuses oleva parameetri *selfLink* väärtuseks, saab vasteks raamatu täieliku andmekaardi (Lisa 1).


```
{
"kind": "books#volumes",
"totalItems": 1,
"items": [{
      "kind": "books#volume",
      "selfLink": "https://www.googleapis.com/books/v1/volumes/kjiloAEACAAJ",
      "volumeInfo": {
            "title": "A Pointless Book",
            "authors": ["Alfie Deyes"],
            "publishedDate": "2014-09",
            "description": "Alfie Deyes is a prominent YouTuber and his
'Pointless' blog has amazing figures: 1.6m subscribers, 5m views per month, 1m
Twitter followers, 650k Instagram and 480k Facebook followers. With all the
humour and quirkiness of Alfie's 'Pointless' YouTube site, this book is packed
with a host of games, pranks, and jokes.",
            "readingModes": {"text": false,"image": false}
}}]}
```


Joonis 8. Google Books API lühendatud tagastatav vastus

3. Rakenduse arendamine Android platvormil

3.1. Android rakenduse struktuur

Android on mobiilne operatsioonisüsteem, mis baseerub Linuxil ning mida arendab Google. Android platvormile loodud rakendused on kirjutatud programmeerimiskeeles Java. Androidi SDK (Software Development Kit - arendustarkvara) tööriistad kompileerivad koodi, ressursid ja andmed ühte APK (Android application package - pakettfaili formaat) faili, mis on käivitatav Android seadmest ning paigaldab rakenduse seadmesse [4].

Android seadmel on paigaldatud rakendus eraldatud turvalises liivakastis:

- Androidi operatsioonisüsteem põhineb Linuxil, millel iga rakendus esindab ühte kasutajakontot.
- Vaikimisi seab operatsioonisüsteem igale rakendusele unikaalse ID ning seadistab rakenduse kõik failid nii, et ainult see kasutajakonto mida konkreetne rakendus esindab pääseb neile ligi.
- Vaikimisi jooksevad kõik rakendused operatsioonisüsteemil eraldi protsessides. Android käivitab protsessi kui mõni rakenduse komponent pannakse tööle ning sulgeb protsessi kui rakendus enam ressursse ei vaja või kui on vaja mälu juurde.
- Igal protsessil on oma virtuaalne masin, millel rakenduse kood jookseb teistest eraldatuna.

Sellisena omab iga rakendus vaikimisi ligipääsu ainult nendele komponentidele süsteemist mida see vajab oma tööks. See loob turvalise keskkonna kus rakendused ei saa ligi pääseda neile süsteemi osadele millele õigused puuduvad. Siiski on olemas võimalused, kuidas üks rakendus saab teiste rakenduste või süsteemi osadega suhelda:

- Rakendus on võimalik seadistada kasutama mingi teise rakenduse või süsteemi osaga sama süsteemi ID-d ning nad pääsevad seega üksteise failidele ligi. Süsteemi ressursside säästmiseks kasutavad Androidis sama ID-ga rakendused ka sama virtuaal masinat, seega mitu rakendust jookseb samas ühtlaselt ligipääsetavas virtuaalruumis.

- Rakendus saab küsida kasutajalt ligipääsu süsteemi osadele nagu näiteks kontaktiraamat, kaamera, välismälu ja paljudele muudele komponentidele. Ligipääsu õigused kinnitatakse rakendust paigaldades.

3.2. Rakenduse komponendid

Iga Android rakenduse komponent on erinev ühendustee süsteemi ja rakenduse vahel. Kokku on neli erinevat tüüpi komponenti, millel igal on erinev eesmärk ja elutsükkel [5]:

- ***Activities*** – *Activity* iseloomustab ühte vaadet kasutajaliidesega. Näiteks uudiste lugemise rakendusel võib olla üks *activity* uudiste listi jaoks ja teine ühe konkreetse uudise lugemiseks. Kuigi kõik *activity-d* töötavad koos on igaüks neist iseseisev üksus ning mingi teine rakendus võib ühe konkreetse *activity* alustada kui see rakendus seda lubab. Nii võib konkreetse näite puhul alustada interneti brauseri rakendus ühe konkreetse uudise lugemise *activity.*
  Iga *activity* on alamklass klassile `Activity.`
- ***Services*** – *Service* ehk teenus on komponent mis töötab rakenduse taustal ja viib läbi pikaajalisi ülesandeid või töötab kaugtöödega. Teenusel ei ole kasutajaliidest. Näiteks teenus võib olla stopper, mis töötab androidi taustal samal ajal kui kasutaja mingit muud rakendust kasutab. Teised komponendid saavad teenuse käivitada või sellega suhelda.
  Iga *service* on alamklass klassile `Service.`
- ***Content providers*** – *Content provider* haldab jagatud osa rakenduse andmetest. Andmeid saab hoida seadme failisüsteemis, SQLite andmebaasis, veebis või mingis muus failide hoidmise hoiustamiseks ettenähtud kohas, kuhu rakendusel on ligipääs. Läbi *content provideri* saavad ka teised rakendused andmetele ligi ja ka neid muuta, kui see on võimaldatud. Näiteks on võimalik ligi pääseda kontaktide andmetele läbi *content provideri* ning kui kasutaja on seda rakendust paigaldades selleks loa andnud.
  Content provider on alamklass klassile `ContentProvider.`
- ***Broadcast receivers*** – *Broadcast receiver* on komponent mis reageerib ülesüsteemsetele teavitustele ja sõnumitele. Paljude allikaks on süsteem ise, näiteks kui ekraan lülitatakse välja või aku tühjenemine. Samuti saavad rakendused neid

teateid saata. Kuigi *boradcast receiver* ei oma kasutajaliidest võib ta luua teavituse süsteemi teavituste ribale ning samal ajal käivitada mõni *activity* või teenus. *Broadcast receiver* on alamklass klassile `BroadcastReceiver.`

3.3. Android rakenduse elutsükkel

Reeglina töötab iga Android rakendus eraldi Linuxi protsessina. Protsess luuakse, kui rakenduse mingit osa käivitatakse ning jääb aktiivseks seniks kuni rakendus enam ei vaja seda ning süsteem saab selle osa mälust endale tagasi, et mõni teine rakendus saaks seda kasutada [5].

Ebatavaline kuid Androidi fundamentaalne osa on see, et rakendus otseselt ise ei otsusta elutsükli pikkuse üle vaid on määratud süsteemi poolt. Nimelt teab süsteem teatud osasid rakendusest mis on aktiivsed ja töötavad, kui olulised need on ja kui palju on süsteemil vaba mälu. Neid andmeid kombineerides määrataksegi iga rakenduse elutsükkel. Kasutades ebakorrektselt rakenduse komponente võib viia selleni, et rakenduse protsess peatatakse, kui rakendus alles töötab.

Et määrata mis protsessid tuleks sulgeda kui süsteemil on vähe mälu, on Androidis loodud teatav hierarhia, et määrata iga protsessi tähtsus süsteemile, mis tuleneb selles protsessis olevatest aktiivsetest komponentidest. Protsessi tüübid on järgmised (tähtsuse järjekorras):

1. **Esiplaani protsess** – on vajalik käesoleva ülesande jaoks. Protsessi peetakse esiplaani protsessiks kui ükskõik milline järgmistest tingimustest on tõene:

   - Protsessis töötab *activity* millega kasutaja hetkel töötab (`onResume ()` meetod on käivitatud).
   - Protsessis on *broadcast receiver* mis on hetkel aktiivne (`BroadcastReceiver.onReceive ()` meetod on käivitunud).
   - Protsessis on teenus mis hetkel töötab (`Service.onCreate ()`, `Service.onStart ()` või `Service.onDestroy ()` meetod on käivitunud).

- Selliseid protsesse on igal aja hetkel süsteemis üpris vähe ning neid peatatakse ainult viimase võimalusena, kui Androidi operatsioonisüsteemi komponentide töötamise jaoks pole piisavalt palju vaba mälu.

2. **Nähtav protsess** – selles protsessis töötab üks *activity*, mis on kasutajale nähtav, kui see ei ole esiplaanis (protsessi `onPause ()` meetod on käivitatud). Selline olukord tekib, kui näiteks esiplaani *activity* käivitab dialoogakna, mille tagant on näha eelmist vaadet.
3. **Teenuse protsess –** selles protsessis töötab teenus mis on käivitatud `startSrvice ()` meetodiga. Need protsessid ei ole kasutajale nähtavad, kuid nad reeglina teevad kasutaja jaoks olulisi asju (näiteks taustal andmete alla laadimine).
4. **Taustaprotsess –** selles protsessis töötab *activity,* mis ei ole hetkel kasutajale nähtav (`onStop ()` meetod on käivitatud). Need protsessid ei mõjuta kasutajakogemust ning neid on võimalik Android süsteemil igal hetkel peatada. Need protsessid on veel omakorda järjestatud tähtsuse järgi, nii et hiljuti kasutatud protsessid sulguksid viimasena, kui on vaja mälu puhastada.
5. **Tühi protsess –** protsess, mis ei hoia endas ühtegi rakenduse komponenti. Need protsessid töötavad kui puhver, et kiirendada rakenduse komponendi käivitumise aega, mis seda protsessi omas.

Rakenduse elutsükkel lõppeb, kui kõikide tema komponentide protsessid on peatatud.

Antud aspekte tuleb kindlasti jälgida rakenduse loomisel komponentide valikul. Antud töö käigus valminud rakendusel oli võimalik lahendada Android süsteemi teavituste kuvamise süsteem ka *Acitity*-na, kuid ainuõige otsus oli see implementeerida teenusena, kuna on vaja käigus hoida pikaajaline protsess, mis teatud intervallide tagant kontrolliks teavituste olemasolu andmebaasist.

3.4. Android rakenduse loomiseks kasutatavad tööriistad

Iga Android rakenduse loomiseks on vajalik Android SDK (Software Development Kit), mis sisuliselt on komplekt Androidi rakenduste loomiseks vajalike arendustööriistu. SDK-sse kuuluvad:

- Vajalikud tarkvara teegid
- Silur ehk *debugger*
- Emulaator
- Androidi programmeerimis liidese (API) kasutamise dokumentatsioon
- Lähtekoodi näidised
- Android rakenduse loomise õpetused

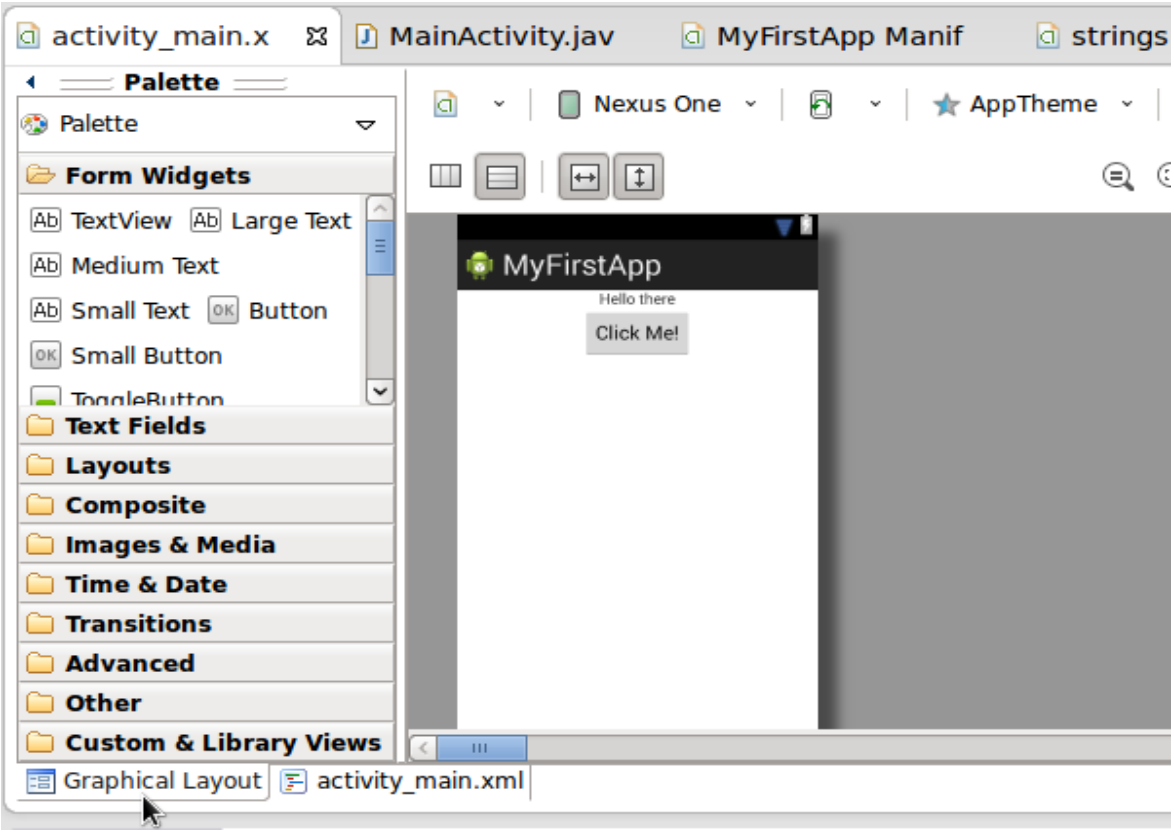
Iga uue Androidi versiooniga uueneb ka SDK. Arendajatel tuleb paigaldada oma arvutile kõige uuemate võimaluste kasutamiseks alati värskeim versioon SDK-st. Kuna Android rakendusi kirjutatakse Javas on vajalik ka JDK (Java Development Kit) olemasolu. Kuigi on võimalik ka üle käsurea Android rakendusi kirjutada SDK olemasolul, siis reeglina tehakse seda mõne arenduskeskkonna abil, millega on SDK integreeritud. Kõige populaarsemad on Eclipse ja Android Studio.

3.4.1 Eclipse

Eclipse[6] esimene versioon on pärit aastast 2001 ning seda kasutatakse peamiselt Java programmide kirjutamiseks. Alates 2009 aastast on võimalik Eclipsega luua ka Android rakendusi. Selleks, et Eclipsega oleks võimalik Android rakendusi kirjutada tuleb paigaldada Google loodud plugin ADT (Android Development Tools), mis sisaldab mitmeid tööriistu, mis võimaldab Android rakenduste loomise Eclipses. Lisaks tavapärastele Eclipse võimalustele on lisandunud XML redaktor, et aidata luua ja muuta Android manifesti, ressursse, menüüsid ja rakenduse visuaalset kavandit (Joonis 9).

Graafilise kuvandi loomise liidesega on võimalik luua rakenduse kasutajaliides ka ilma XML struktuurkoodi kirjutamata. Võimalik on *drag and drop* meetodil vajalikud kasutajaliidese elemendid lõuendile lohistada ja nende erinevaid parameetreid muuta ja koheselt on tulemus kuvatud kasutajale ilma rakendust kompileerimata.Samuti on lisatud indekseerimine ja kiire viitamine erinevatele Android ressurssidele, mida on programmikoodis kasutatud.

[6] https://eclipse.org/



Joonis 9. ADT graafilise kuvandi loomise liides

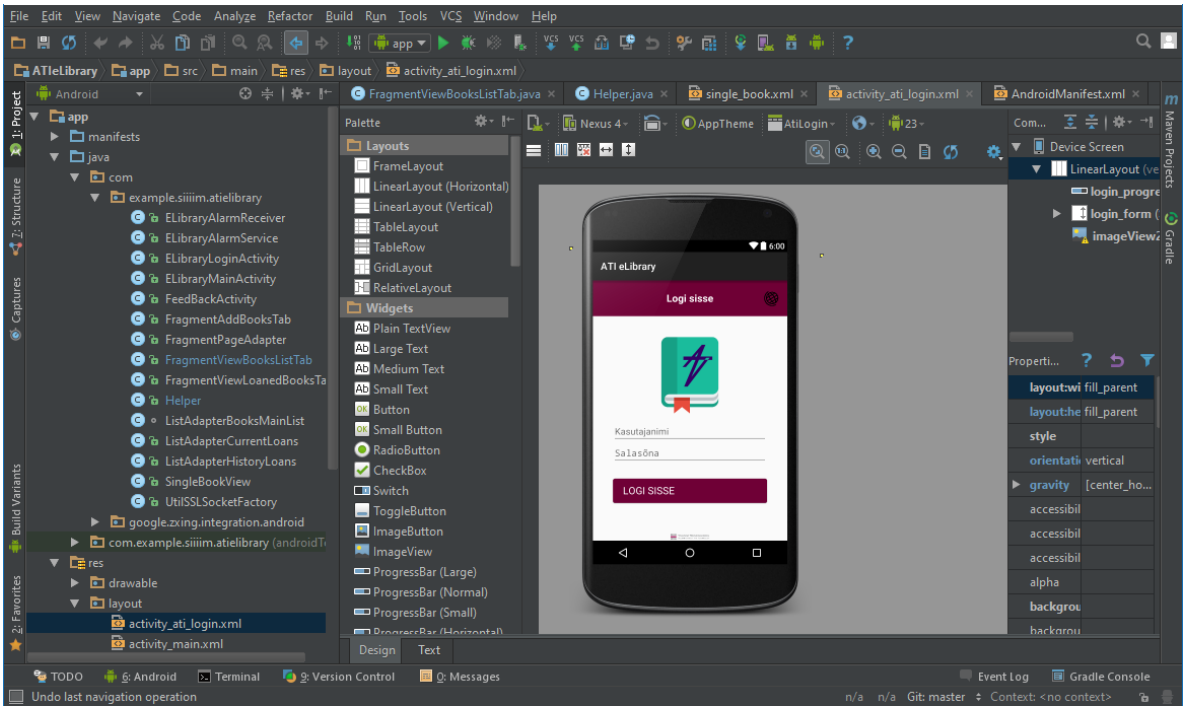
Eclipse koos ADT-ga oli soovitatud tööriist Android rakenduste arendamiseks kuni 2015. aastani, millal tuli välja JetBrains[7] poolt toodetud Android Studio. Kuna Eclipse ei ole enam toetatud Android arendusplatvorm, soovitab Google kõigil Eclipse kasutajatel oma Android projektid migreerida Android Studio-sse üle.

3.4.2 Android Studio

2013. aastal Google I/O-l välja kuulutatud Android Studio[8] on alates 2015 aastast soovitatud tööriist Android rakenduste loomiseks ning on konkreetselt Android jaoks loodud ja kohandatud. Kasutajaliidese (Joonis 10) poolest on arenduskeskond sarnane kõikide teiste JetBrains toodetega, nagu näiteks Intellij[9] ja WebStorm[10]. Android Studio omab kõiki tuttavaid võimalusi mis olid ka Eclipsel, kuid neid on oluliselt täiendatud. Parandatud on

[7] https://www.jetbrains.com/
[8] http://developer.android.com/sdk/index.html
[9] https://www.jetbrains.com/idea/
[10] https://www.jetbrains.com/webstorm/

Android koodi refaktoreerimist ja automaatset koodilõpetamist. Mugavalt on võimalik projekt siduda versioonihaldussüsteemiga ning loodud on ka integratsioon GitHubiga[11]. Lisatud on ka võimalus visuaalsel kuvandi redaktoril kuvada mitut vaadet korraga.



Joonis 10. Android Studio arenduskeskond

Antud lõputöös valminud Android rakenduse prototüübi arendamiseks kasutati Android Studiot.

[11] https://github.com/

4. Arvutitehnika instituudi digitaalse raamatukogu veebiteenus

4.1. Veebiteenuse tutvustus

Antud töö käigus valminud veebiteenus oli üks Arvutitehnika instituudi raamatukogu Android mobiilirakenduse loomiseks vajalik mittefunktsionaalne nõue. Kuna rakenduse tarbeks olevad andmed on suures osas kõikide kasutajate vahel jagatud, on vajalik tsentraalne andmebaas. Oma lihtsa seadistamise ja olemasolu tõttu sai valitud MySQL andmebaasi haldamise süsteem. Selleks, et kasutaja tegevused rakenduses jõuaksid andmebaasini on vajalik vahekihina veebiteenus, sest otse Android süsteemist pole võimalik MySQL andmebaasi jaoks päringuid käivitada. Samuti oli piirang Arvutitehnika instituudi ja laiemalt terve Infotehnoloogia teaduskonna veebiserveritele ligipääsevate seadmete ja IP-de osas.

Vahekihina loodud veebiteenus töötab Apache veebiserveris ning on loodud kasutades programmeerimiskeelt PHP. Kuna veebiteenus ja rakenduse andmebaas töötavad samas serveris, siis see eemaldab ligipääsu piirangud. Selleks, et PHP-s kirjutatud MySQL päringut käivitada on vaja veel PHP jaoks laiendust MySQLi, mis tegeleb SQL päringute töötlusega PHP rakenduses.

Android rakenduse tarbeks on võimalik sooritada veebiteenusega alljärgnevaid operatsioone:

- Rakendusse sisse logimine
- Uue raamatu lisamine andmebaasi
- Olemasolevate raamatute küsimine andmebaasist
- Raamatute laenutamine ja tagastamine
- Laenutuste ajaloo kuvamine
- Tagasiside jätmine

Lisaks tavapärasele veebiteenusele töötab serveris ka teavituste genereerimise süsteem, mis aktiveerub iga tund, ning genereerib nendele kasutajatele, kelle raamatu tagastamistähtaeg hakkab lähenema, süsteemse teavituse. Samal ajal töötab ka Android rakenduses teenus, mis iga nelja tunni tagant kontrollib uute teavituste olemasolu ning kuvab neid automaatselt telefonis, ka siis kui rakendust samal ajal ei kasutata.

4.2. Veebiteenuse arhitektuur ja kasutusjuhend

Loodud veebiteenus on REST tüüpi ning lubatud on nelja tüüpi operatsioone – *login, insert, select* ja *update*.

Operatsioon antakse kaasa URL-le GET argumendina: `/?method=select` Ülejäänud päring tuleb kaasa anda POST argumentidena. Vajalikud on kolm POST argumenti:

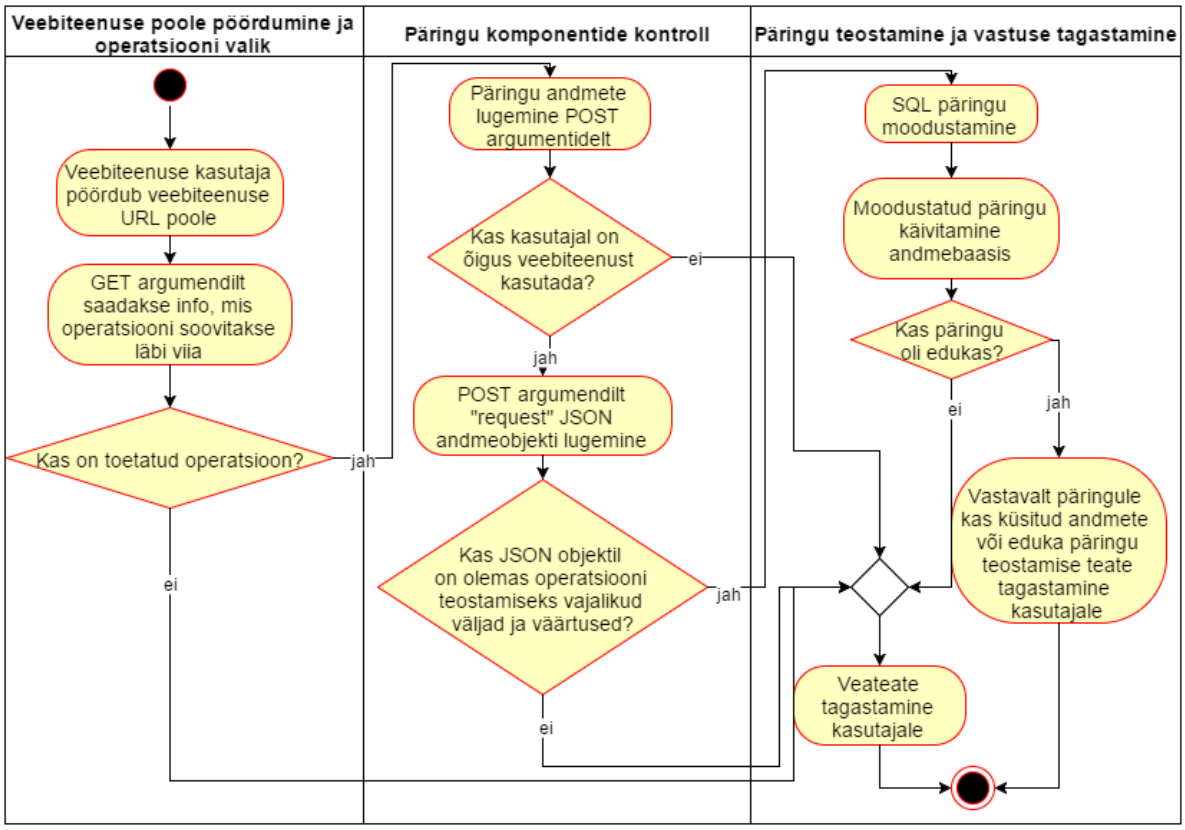
- *user_id*, täisarvuline muutuja, rakendust kasutava kasutajakonto unikaalne identifikaator
- *key*, string tüüpi muutuja, räsifunktsiooniga genereeritud tähtede jada, tagab ühenduse turvalisuse
- *request*, string tüüpi muutuja, sisaldab endas JSON kujul täpseid andmeid päringu vormistamiseks

Erandiks on vaid *login* operatsioon kus pole vaja *request* argumendi kaasa andmist.

Veebiteenus on loodud võimalusega, et teda saab kasutada andmebaasis suvalise tabeliga lubatud operatsioonide läbiviimiseks. Süsteemselt pole piiratud milliste tabelitega ühendust võtta on võimalik. Eelduseks on see, et veebiteenust kasutab professionaalne ning dokumentatsiooniga tutvunud programmeerija. Päringu sisuosa on paigutatud POST argumentidesse, millest *key* on reaalajas muutuv räsifunktsiooniga genereeritud kasutajapõhine unikaalne võti.

Veebiteenuse kasutamiseks (Joonis 11) peab igal ühenduval kasutajal olema andmebaasis unikaalne kasutajakirje. Läbi Arvutitehnika instituudi raamatukogu Android rakenduse ühendumisel luuakse igale ITA (IT Akadeemia - koostööprogramm ja kaubamärk) kasutajakontot omavale isikule ka antud veebiteenuse ligipääs. Juhul, kui veebiteenuse kasutaja teostab operatsiooni mis pole toetatud või lubatud kuvatakse talle vastusena veateadet (Lisa 3. Vastus 4).

Kui väljad *user_id* ja *key* kannavad endas ainult ühte lihtväärtust siis väljal *request* hoitakse JSON andmeobjekti (Joonis 12). Ainuke kohustuslik JSON parameeter on *table* (Lisa 3. Päring 1).



Joonis 11. Veebiteenuse kasutamise tegevusdiagramm

Samuti on võimalik ka päringu tulemusi kitsendada, andes tingimused kaasa JSON objekti väljale *where* (Lisa 3. Päring 2). Kitsendusi võib olla mitmeid ning nad on kõik korrutuvad ehk *and* tüüpi. Samuti on võimalik tulemuste järjestust ning grupeerimist määrata JSON objekti väljadega *group* ja *order* (Lisa 3. Päring 3).

Viimase *select* operatsiooni lisavõimalusena on funktsioon kombineerida seotud väljade kaudu ühe andmebaasi tabeli tulemusi teise tabeli kirjetega. SQL liitmisfunktsioonidega ehk *joinidega* on võimalik sooritada kõiki MySQL-s toetatud *joine.* Liitmistingimusi võib olla rohkem kui üks ning lisanduvad tingimused tuleb kaas anda *join* massiivi elemendi väljale *where* (Lisa 3. Päring 4).

Select operatsiooni tagastus on JSON kujul ning sisaldab endas vastuseks saadud kirjete arvu ja JSON massiivina kõiki kirjeid eraldi. Veebiteenuses on *select* päring koostatud selliselt, et andmebaasi tabelist küsitakse kõikide tulpade väärtused (Lisa 3. Vastus 3).

Insert ja *update* operatsioonid on päringu poole pealt üpris sarnased. Ainuke erinevus on, et *update* puhul on JSON parameeter *where* kohustuslik. Nii *insert* kui ka *update* puhul on kohustuslik JSON parameeter *body.* Sellele väljale lähevad uue või uuendatava kirje kõik omadused. Väljad peavad olema üksüheses vastavuses andmebaasis oleva tabeli tulpadega (Lisa 3. Päring 5). Nii *insert* kui ka *update* operatsiooni korral tagastab teenus eduka päringu korral vastuse (Lisa 3. Vastus 1, Lisa 3. Vastus 2). Samuti tagastatakse vigade puhul veateade detailise vea põhjusega.

Kolme baasoperatsiooniga on võimalik tegutseda suvalise MySQL andmebaasiga. JSON andmeobjektide lihtsuse tõttu on ka teenuse kasutajatel üpris mugav ja lihtne päringute formaat (Joonis 12) ning sisu moodustada. REST arhitektuurile omapäraselt on iga operatsiooniks vajaminev päring täpselt nii pikk, kui hetkel vaja. Päring ning ka selle vastus sisaldab ainult seda informatsiooni, mida on kaasa antud või küsitud veebiteenuselt. Teenus ei ole ühe rakenduse keskne, vaid võimaldab mitmetel erinevatel tarbijatel mistahes eesmärgil seda kasutada. Eelnevalt on vaja täiendada andmebaasi vaid vajalike andmetabelitega.

```
{       table: tabeli_nimi,
        where: {        tulba_nimi:tulba_väärtus,
                        ....
                },
        group: tulba_nimi,
        order: tulba_nimi,
        join: [
                        {       type:liitmisoperatsiooni_tüüp,
                                table:liidetav_tabel,
                                column1:põhitabeli_liidetav_tulp,
                                column2:liidetava_tabeli_tulp,
                                where: {
                                        tabeli_nimi.tulba_nimi:tulba_väärtus,
                                        ...
                                }
                        },
                        ...
                ],
        body:{          tulba_nimi:tulba_väärtus,
                        ....
                }
}
```

Joonis 12. Raamatukogu veebiteenuse päringu request parameetri JSON objekti struktuur

5. Arvutitehnika instituudi digitaalse raamatukogu Android rakendus

5.1. Rakenduse tutvustus ja eesmärk

Käesoleva töö raames valminud Android rakenduse prototüübi eesmärk on parendada Arvutitehnika instituudi kirjanduse kasutamist ning tekitada ülevaade olemasolevast raamatutebaasist.

Loodud Android mobiilirakenduse nimi on „ATI eLibrary“ ning omab järgmisi funktsioone:

- Sisse logimine ja rakenduse kasutamine olemasoleva ITA infosüsteemi kontoga
- Raamatute lisamine andmebaasi
- Raamatute lisamisel vajaliku ISBN koodi skaneerimine nutitelefoni kaameraga raamatu vöötribalt
- ISBN koodi alusel raamatu detailse informatsiooni pärimine Google Books veebiteenuselt
- Kõikide olemasolevate raamatute kuvamine nimekirjana ning otsing nende seast
- Välja laenutatud raamatutel vastava märgistuse kuvamine
- Iga raamatu detailse informatsiooni ja asukoha kuvamine
- Raamatute laenutamine mingiks kindlaks perioodiks ning raamatute tagastamine
- Sisse logitud kasutajale aktiivsete laenutuste ning laenutuste ajaloo kuvamine
- Teavituste ehk *notification*-te saatmine Android seadmes kui laenutuse tähtaeg hakkab lähenema
- Rakenduse kasutamine nii eesti kui ka inglise keeles

„ATI eLibrary“ on valmistatud töötama Android versioonil 4.4 „KitKat“ ning edukaks paigaldamiseks on vaja Android seadme seadetest võimaldada teadmata allikatest pärinevate rakenduste paigaldamine. Samuti peab iga rakenduse kasutaja omama ITA keskkonna kontot.

5.2. Rakenduse kasutusjuhud

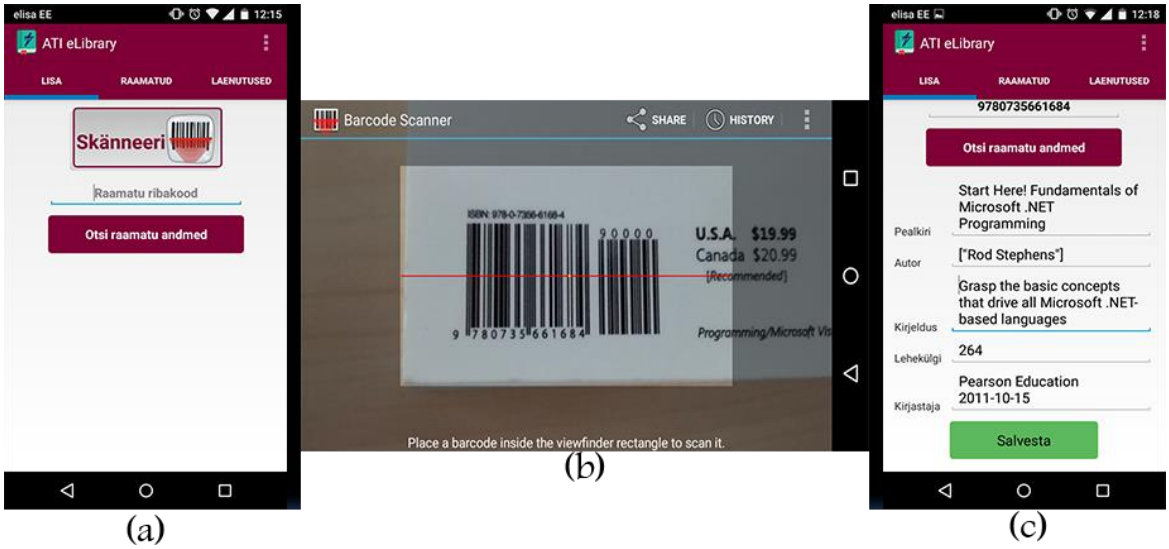
Loodud rakenduse peamine eesmärk on luua Arvutitehnika instituudile ülevaade olemasolevast õppekirjandusest ning võimaldada otsida endale sobivat raamatut ning seda laenutada.

Raamatutebaasist ülevaate loomiseks tuleb esmalt olemasolev kirjandus sisestada andmebaasi (Joonis 13.a). Selleks tuleb avada rakendus ning navigeeruda „Lisa“ vaatesse kasutades ülemist menüüriba. Järgevalt on võimalus skaneerida nutitelefoni kaameraga raamatu ribakood, millel on raamatu ISBN. Esmakordsel skaneerimisel on vajalik paigaldada rakendus nimega „Barcode Scanner“[12], mille saab allalaadida Google Play[13] rakenduste poest. Seejärel tuleb vastavalt kuvatud juhistele asetada raamatu ribakood kaamera vaatevälja ning rakendus skaneerib automaatselt ribakoodi (Joonis 13.b). Peale ribakoodi skaneerimist suunatakse kasutaja tagasi ATI eLibrary raamatute lisamise vaatesse ning tuleb vajutada nupule „Otsi raamatu andmed“. Seejärel teostab rakendus otsingu Google Books[14] veebiteenusega raamatu andmete kohta otsingu. Juhul kui midagi leitakse, siis täidetakse andmeväljad automaatselt. Peale seda on kasutajal võimalus modifitseerida lisatava raamatu andmeid ning salvestada kirje andmebaasi (Joonis 13.c). Raamatu haldajaks saab automaatselt raamatu lisaja.

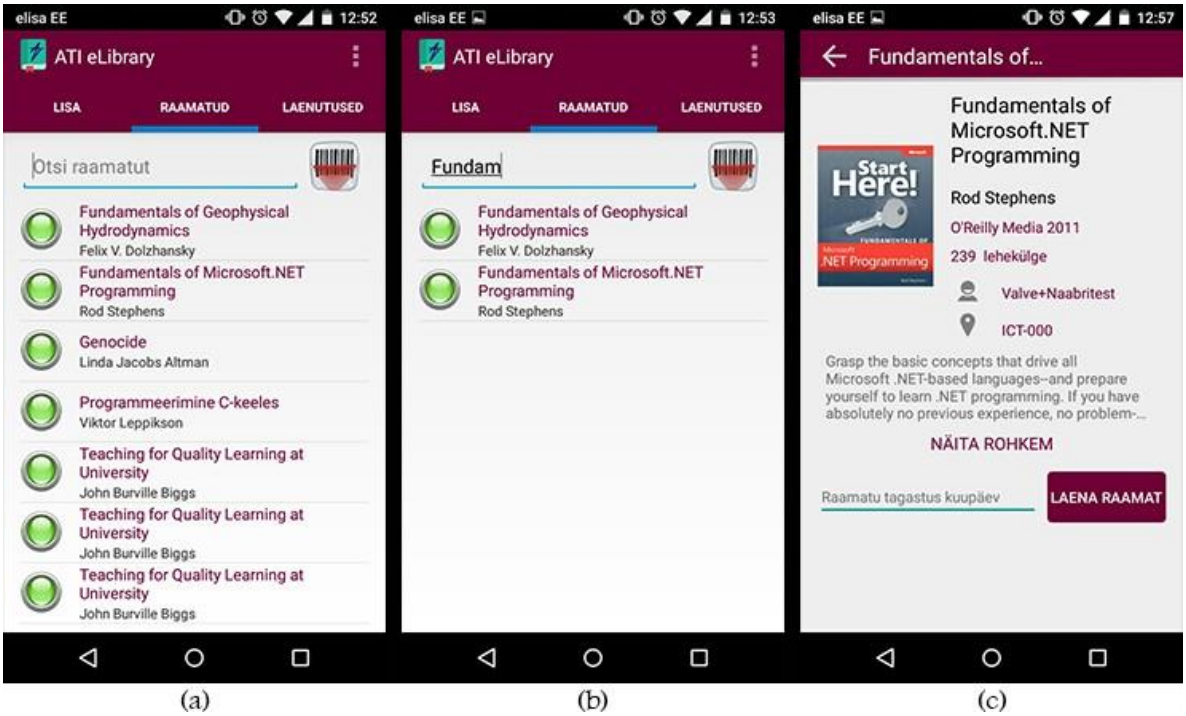
Andmebaasi lisatud raamatud on kõigile nähtavad ja neid saab vaadata rakenduse „Raamatud“ vaates milleni saab liikuda ülemise menüüriba kaudu. Selles vaates on nimekirjas kõik raamatud ning kuvatud on nende staatus, raamatu pealkiri ja autor (Joonis 14.a). Rohelise ikooniga on tähistatud raamatud mis on saadaval ning punasega need mis on välja laenutatud. Kasutada on võimalik ka otsingut, mille abil on võimalik tulemusi piirata (Joonis 14.b). Otsida on võimalik raamatu nime, autori ja ISBN koodi alusel. Samuti on võimalik otsida kasutades ribakoodi skaneerimist. Selleks tuleb vajutada ribakoodi skaneerimise nupule ja nutitelefoni kaameraga skaneerida ribakood. Juhul kui skaneeritud ribakoodiga raamatuid esineb andmebaasis, siis on need nimekirja kuvatud. Vajutades

[12] https://play.google.com/store/apps/details?id=com.google.zxing.client.android
[13] https://play.google.com
[14] https://books.google.com/books

nimekirjas raamatu rea peale kuvatakse raamatu detailsem info, kaasa arvatud raamatu omanik ja asukoht (Joonis 14.c).



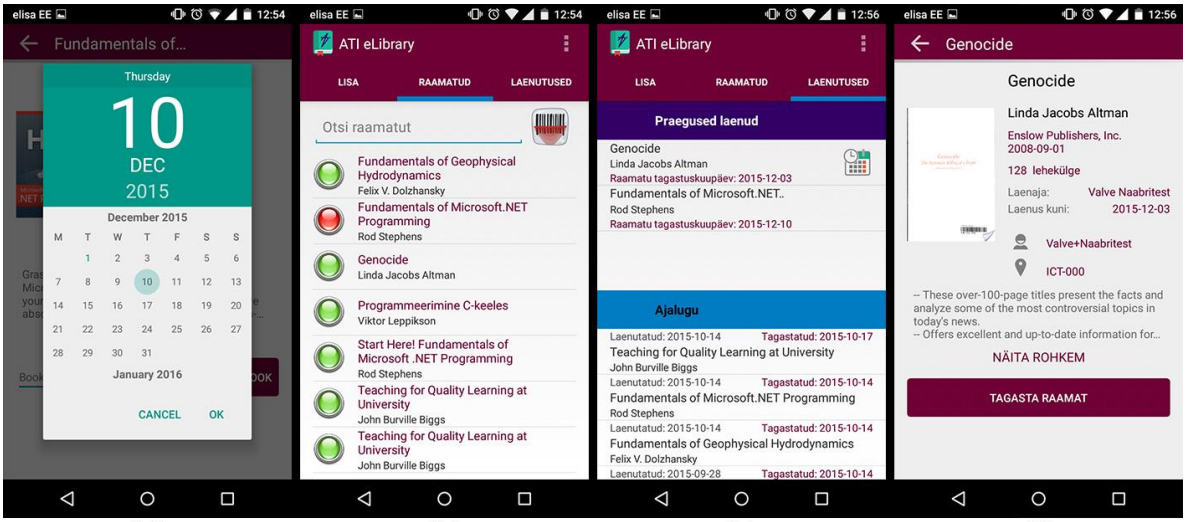
Joonis 13. Rakendusega ATI eLibrary raamatu lisamine



Joonis 14. Rakenduse ATI eLibrary raamatute nimekiri ja raamatu vaade

Raamatute laenutamiseks tuleb raamatute nimekirjast vajutada soovitud raamatu peale, mille staatuse ikoon on roheline (Joonis 15.b). Seejärel tuleb määrata raamatu detailses vaates tagastus kuupäev . Seda saab teha vajutades tekstile „Raamatu tagastus kuupäev“ ning valida avanenud hüpikaknas kuupäev milleni soovitakse raamatud laenutada (Joonis 15.a). Tagastus kuupäev on päeva täpsusega. Peale kuupäeva valimist ning selle kinnitamist tuleb vajutada nupule „Laena raamat“, mis teostab raamatu laenamise operatsiooni. Minnes tagasi raamatute nimekirja on laenatud raamatu staatuse ikoon muutnud punaseks ning sellisena on see raamat kuvatud ka teistele kasutajatele.

Kasutajatel on võimalik näha kõiki nende laenutusi „Laenutused“ vaatest, milleni saab liikuda ülemisest ribamenüüst (Joonis 15.c). Vaade on jagatud kaheks, milles ülemisel poolel on kuvatud hetkel aktiivsed laenud ja alumisel on näha laenutuste ajalugu. Aktiivsete laenude nimekirjas on raamatutele, mille tagatuskuupäev on peatselt lähenemas kuvatud kalendri ikoon. Igale reale on võimalik vajutada mille peale kuvatakse raamatu detailne vaade. Raamatu tagastamiseks tuleb laenutatud raamatu detailsesse vaatesse minna ning seejärel vajutada nupule „Tagasta raamat“ (Joonis 15.d). Peale seda operatsiooni on raamat tagastatud ning jälle kõikidele teistele kasutajatele saadaval.

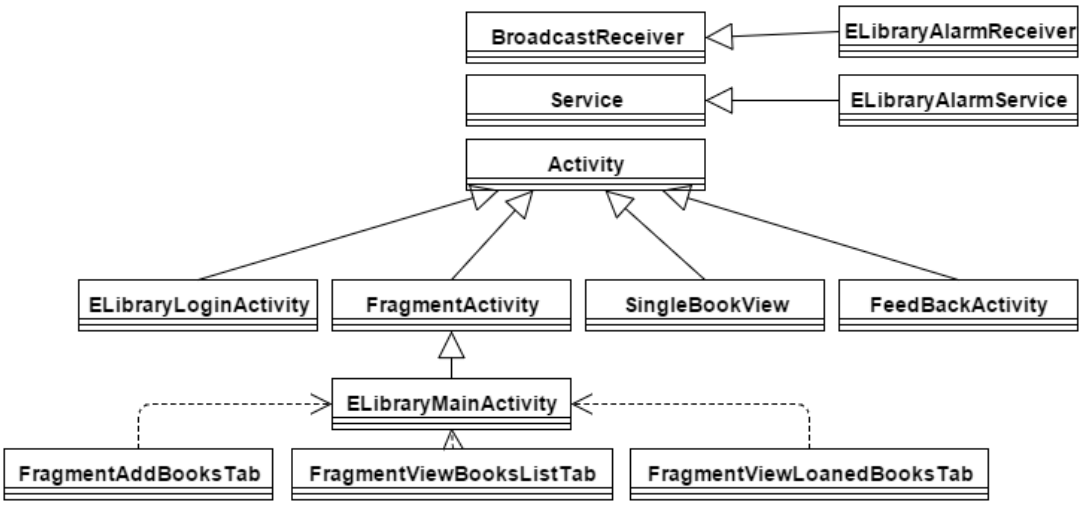


Joonis 15. Rakenduses ATI eLibrary raamatute laenutamine

5.3. Rakenduse struktuur

Rakendus ATI eLibrary siht Android versioon on 5.0, kuid see töötab ka versiooniga 4.2 või uuemaga. See tähendab seda, et rakendus on loodud ja testitud versiooniga 5.0 kuid ei sisalda endas komponente, mis ei tööta versioonil 4.2 või uuem.

Rakendus sisaldab endas nelja *Acitivity-t,* ühte *Service-t* ehk teenust ja ühte *Broadcast Receiver-t* (Joonis 16). `ElibraryLoginActivity` on rakenduse esmane *activity,* mis käivitub koos rakendusega. See sisaldab endas rakendusse sisse logimise vaadet. Sisse logimine tehakse kasutades ITA siseveebi kasutajakontot ja läbi ITA autentimisteenuse. Autentimine nõuab internetiühenduse olemasolu ning teostatakse asünkroonselt kasutades klassi mis laiendab `AsyncTask` klassi. Nii ei hangu kogu *activity* autentimise ajaks ära ning on endiselt interaktiivne.



Joonis 16. Rakenduse ATI eLibrary struktuur

Rakenduse põhiline vaade, mis sisaldab endas raamatute lisamise, nimekirja ja laenude vaadet on üks neljast *Activity-st* ja see on laiendus `FragmentActivity` klassile mis on omakorda laiendus `Activity` klassist. `FragmentActivity` klassi laiendades on võimalik jagada üks *Activity* mitmeks osaks ehk ATI eLibrary puhul kolmeks klassiks mis laiendavad `Fragment` klassi – `FragmentAddBooksTab`, `FragmentViewBooksListTab` ja `FragmentViewLoanedBooksTab`. `FragmentActivity` võimaldab vaadete (fragmentide) vahel liikuda kas vajutades *tab*-i nimele või vaadete vahel nutitelefoni puutetundlikul ekraanil horisontaalselt lohistades.

Raamatute lisamise fragmendi (Joonis 13) ülesandeks on kuvada raamatute lisamise vorm. Skaneerimise nupuga käivitatakse „Barcode Scanner“ rakenduse ribakoodi skaneerimise funktsioon. See on saavutatud läbi Zxing[15] klassiteegiga mis võimaldab teises rakenduses käivitada skaneerimise *acitivity*. Peale tulemuse saamist tagastatakse ribakoodi väärtus ATI eLibrary-sse ja „Barcode Scanner“ rakendus sulgetakse. Ribakood tagastatakse fragmendi ribakoodi lahtrisse, kuhu on võimalik see ka käsitsi sisestada kasutades telefoni klaviatuuri. Peale ribakoodi sisestamist on võimalik käivitada päring mis küsib Google Books API-lt sellise ISBN koodiga raamatu kohta andmeid. Selle jaoks käivitatakse samuti asünkroonselt päring kasutades klassi `UtilConnectionManagerRequest.` Peale päringu läbiviimist tulevad nähtavale raamatu andmete lisamiseks või Google Books API-st saadud andmete muutmiseks vajalikud lahtrid. Seejärel on võimalik uus raamat salvestada andmebaasi kasutades Arvutitehnika instituudi raamatukogu platvormi veebiteenuse *insert* operatsiooni. Raamatu lisamiseks vajalik päring veebiteenusega teostatakse samuti asünkroonselt taustal.

Nimekirja fragmendis (Joonis 14.a) kuvatavad andmed saadakse Arvutitehnika instituudi raamatukogu platvormi veebiteenuse *select* operatsiooniga, milles küsitakse kõik *books* tabelis olevad raamatud ning liidetakse tulemusi *books_loaned* tabeliga, et saada teada raamatu staatus. Iga tulemuse rida kuvatakse fragmendi välja *ListView* ühe elemendina. Kogu vaade asub visuaalse elemendi `SwipeRefreshLayout` sees mis võimaldab vaadet vertikaalselt lohistades aktiveerida mingi tegevus. Antud fragmendi puhul aktiveeritakse vaate värskendamine ehk kõik tulemused laetakse andmebaasist uuesti ning juhul kui teised kasutajad on samal ajal lisanud uusi raamatuid, siis need ilmuvad nüüd nimekirja.

Kolmandas fragmendis (Joonis 15.c) on kuvatud aktiivsed laenutused ja ka laenude ajalugu. Tehniliselt on see vaade sarnane raamatute nimekirja vaatega, kuid ühe *listview* asemel on kaks. Samuti teostatakse kaks erinevat päringut – mõlema *listview* jaoks üks. Samuti asub terve fragment visuaalse elemendi `SwipeRefreshLayout` sees ning vertikaalse *swipe* puhul värskendatakse vaade.

Vajutades kummagi fragmendi *listview* elemendile avatakse raamatu vaate klass `SingleBookView` ehk kolmas *Activity* (Joonis 14.c). Vaadet initsialiseerides tehakse

[15] https://github.com/zxing/zxing

taaskord päring Arvutitehnika instituudi raamatukogu platvormi veebiteenuse pihta ning päritakse raamatu detailsed andmed *books* tabelist. Kuna üheks kuvatavaks elemendiks on raamatu kaane illustratsioon ning andmebaasis on selle viideks ainult URL, siis tuleb pildi saamiseks teha lisanduv päring. Vaates on võimalik teostada 2 operatsiooni olevalt raamatu staatusest ja kasutajast, kes on sisse logitud – raamatu laenamine ja raamatu tagastamine. Neist viimane on võimalik ainult siis, kui parasjagu sisse logitud kasutaja on ka raamatu laenutaja. Raamatu laenamise jaoks tehakse raamatukogu platvormi veebiteenusega *insert* operatsioon ning tagastamisega *update*.

Neljas ja viimane *activity* on rakenduse info vaade, milles on lühike kirjeldus rakendusest ning sealt on võimalik ka saata rakenduse kohta tagasisidet kasutades Arvutitehnika instituudi raamatukogu platvormi veebiteenuse *insert* operatsiooni.

Rakenduse ainukese *service* ehk teenuse `ELibraryAlarmService` eesmärk on iga nelja tunni tagant teha raamatukogu platvormi veebiteenusega päring millega vaadatakse kas rakenduse kasutajale on genereeritud lähenevate laenutuste tagastuskuupäevade tõttu mõni teavitus. Antud eesmärgiks on vaja kindlasti kasutada just teenust, sest vastasel juhul Android operatsioonisüsteem sulgeb rakenduse peale mõnda aega kui rakendust pole aktiivselt kasutatud. Sellisest käitumisest on täpsemalt kirjutatud peatükis 3.3. Teavitusi genereerib veebiserveris töötav *cron job*. Juhul kui mõni teavitus on andmebaasi genereeritud, siis *BroadcastReceiver*-ga `ElibraryAlarmReceiver` luuakse Android seadmes teavitus ning kuvatakse see kasutajani.

Üle terve rakenduse on kõikide klasside jaoks vajalikud ka Android seadmesse salvestatud andmed, mis antud juhul on *Shared Preferences* ehk jagatud eelistuste kujul. Andmed on salvestatud `SharedPreference` klassi objektil millel on talletatud kasutaja identifikaator, nimi, kasutaja kabineti number, Arvutitehnika instituudi raamatukogu platvormi veebiteenuse võti ja kasutaja valitud keel, millega ta rakendust kasutab. Andmed on salvestatud sellepärast, et oleks võimalik peale esialgset rakendusse sisse logimist järgmistel kordadel seda koheselt kasutama hakata ilma sisse logimise protsessita. `SharedPreference` objekti sisu hoitakse kuni kasutaja otsustab rakendusest välja logida

mille tagajärjel objekti sisu hävitatakse. Kuna antud juhul on ATI eLibrary `SharedPreference` nähtav ainult selle rakenduse piires on ka andmed turvalised.

5.4. Rakenduse liidestamine veebiteenustega ja HTTP päringute teostamine

ATI eLibrary rakendus kasutab erinevate ülesannete läbiviimiseks kolme veebiteenust:

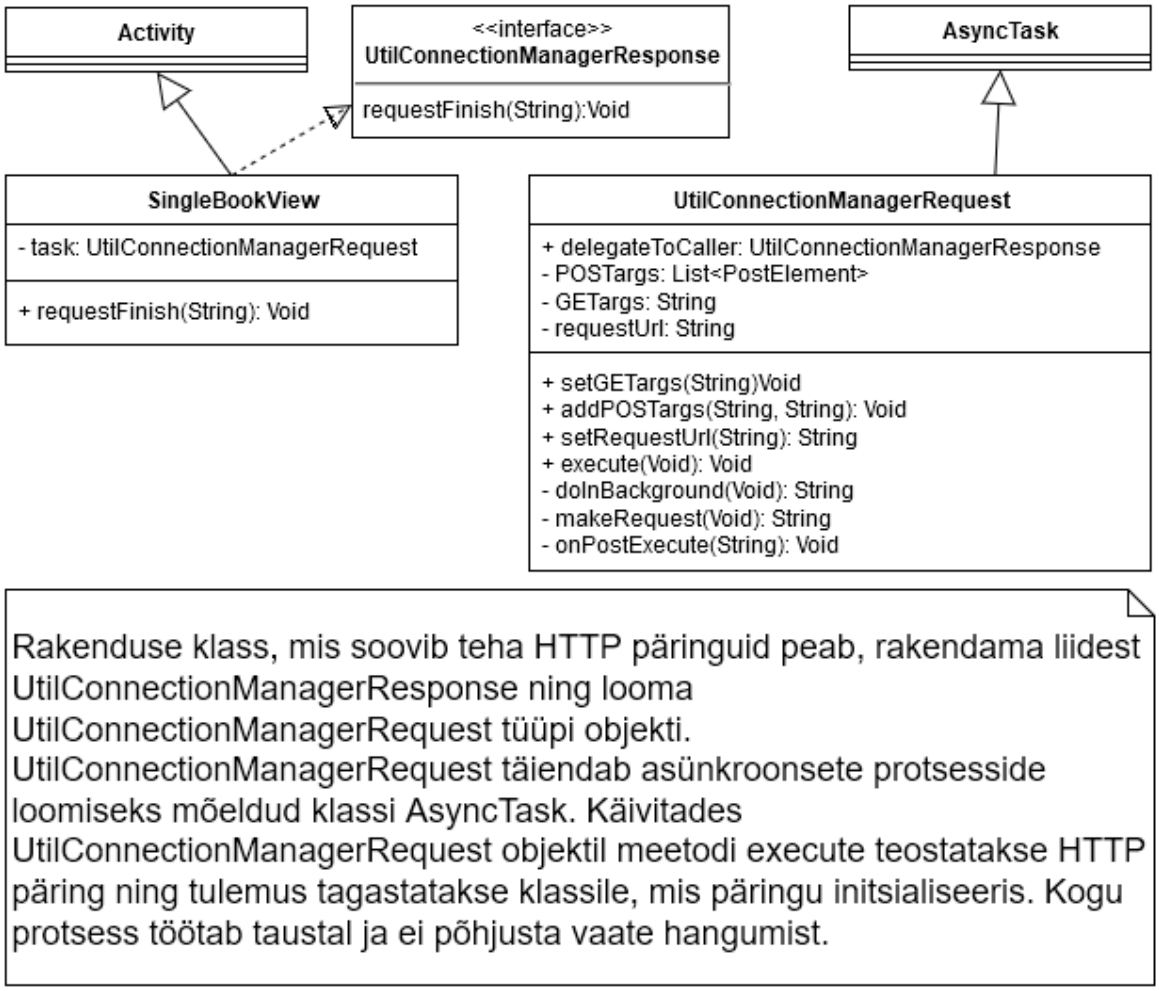
- ITA autentimismoodul
- Arvutitehnika instituudi raamatukogu platvormi veebiteenus
- Google Books API

Kõik kolm teenust põhinevad REST tüüpi arhitektuuril ning rakenduse tasandil pöördutakse kõigi nende poole sama klassiga – `UtilConnectionManagerRequest.` Arvutitehnika instituudi raamatukogu platvormi veebiteenust kirjeldab peatükk 4 ja Google Books API-t peatükk 2.5.

ITA autentimismooduli päring koosneb nii GET kui ka POST parameetritest. Sarnaselt Arvutitehnika instituudi raamatukogu platvormi veebiteenusele on POST argumendiks JSON andmeobjekt. Turvalisus tagatakse väärtuste krüpteerimisega räsifunktsiooniga. Samuti on tagastus JSON formaadis, ning sisaldab endas informatsiooni, kas päring oli edukas ning sisse logitud kasutaja andmeid, mida on Android rakenduses vaja. Autentimismooduli poole pöördutakse ainult rakendusse sisse logimisel. Edaspidised rakenduse käivitamised, kui vahepeal pole välja logitud, ei nõua lisanduvat autentimist, sest rakenduse `SharedPreference` objektile on talletatud sisse logitud kasutaja informatsioon.

Jätkusuutlikuks ja optimaalseks veebiteenustega liidestamiseks oli vaja luua dünaamiline rakenduse komponent mis suudaks suhelda erinevate veebiteenustega, kuid mida ei peaks liigselt dubleerima. Klassi `UtilConnectionManagerRequest` on võimalik kasutada kõikidel klassidel mis on implementeerinud liidese `UtilConnectionManagerResponse,` mis tegeleb läbiviidud päringu tulemuse tagastamisega selle initsialiseerinud klassini (Joonis 17).

Klass mis soovib mingit päringut teostada, peab looma `UtilConnectionManagerRequest` objekti, paika panema nõutud parameetrid ja väärtustama POST või GET argumendid ning kuna `UtilConnectionManagerRequest` laiendab klassi `AsyncTask` saab päringu käivitada meetodiga `execute()`. Seejärel teostatakse varem loodud objekti `doInBackground()` meetodis defineeritu, milleks on `makeRequest()` meetod (Lisa 4), kus viiakse läbi päring. Tulemus tagastatakse objekti `onPostExecute()` meetodisse, mis omakorda käivitab päringut teostava klassi liideses `UtilConnectionManagerResponse` nõutud meetodi `requestFinish()`, kus viiakse andmetega just selle vaate jaoks vajalik operatsioon läbi.



Joonis 17. Rakenduses ATI eLibrary HTTP päringute teostamiseks vajalikud klassid SingleBookView näitel

Selliselt on võimalik teostada suvalist HTTP päringut ilma lisaloogikat lisamata. Ainuke piirang on see, et vastus peab olema JSON andmeobjekt.

6. Arvutitehnika instituudi digitaalse raamatukogu platvormi analüüs

6.1. Platvormi rakendamine ja tagasiside kasutajatelt

Platvormi kasutuselevõtu lihtsustamiseks on loodud liidestus ITA siseveebiga, seega ei nõua Android rakenduse ATI eLibrary kasutamine seda, et kasutajad looksid endale uued kasutajakontod – kasutada saab samade andmetega millega siseveebi logitakse. Samuti on arvestatud sellega, et enamus Android seadmete kasutajaid on endiselt versioonil 4.2 või uuem (Lisa 5). Sellepärast on rakenduse minimaalne versioon seatud 4.2 peale.

Raamatukogu platvorm ise koosneb kolmest eraldi seisvasvast komponendist – veebiteenus, andmebaas ja Android rakendus. Andmebaasi ja veebiteenuse migreerimine Arvutitehnika instituudi serverisse sisaldab endas ainult vastava andmebaasi loomist ning veebiteenuse koodibaasi ümber kopeerimist. Andmebaasi saab lihtsalt ja kiirelt luua kasutades arenduskeskkonna koopiat. Veebiteenuses tuleb muuta vaid `Constants.php` failis andmebaasi poole pöördumiseks vajaliku MySQL-i kasutajakonto andmeid. Samuti tuleb lisada veebiserverile täiendav *cron job* mis pöörduks iga tund veebiteenuse URL poole aga oleks kataloogis `notifications` olevale failile `index.php` suunatud.

ATI eLibrary rakenduse lähtekoodis tuleb samuti vastavalt veebiteenuse aadressile kohandada `Constants.java` failis asuvat veebiteenuse URL-i. Peale seda on vajalik rakenduse uuesti kompileerimine ja .apk faili *buildimine* ehk ehitamine. Kuna loodud rakendus on momendil rangelt ainult Arvutitehnika instituudile suunatud, siis ei ole võimalik lisada rakendust Google Play rakenduste poodi. Selle asemel tuleb rakenduse paigaldamiseks ettenähtu .apk fail manuaalselt telefoni lisada ning seejärel käivitada.

Rakenduse funktsionaalsust ja kasutajamugavust testis Arvutitehnika instituudi nooremteadur Siavoosh Payandeh Azad, kes on raamatukogu platvormi idee autor. Saadud tagasisides mainib Azad, et rakenduse olemasolev funktsionaalsus töötab hästi, kuid sooviks näha veel täiendavaid võimalusi nagu näiteks raamatutele siltide lisamine ja raamatu andmete modifitseerimine läbi rakenduse, peale selle lisamist andmebaasi.

6.2. Infosüsteemi analüüs ning puudused

ATI eLibrary Android rakenduse üks nõudmistest oli kasutajate põhine tegevus. Selleks oli vaja võimaldada igal rakenduse kasutajal rakendusse sisse logida. Parema kasutuskogemuse tagamiseks ei nõua rakendus uue kasutajakonto loomist vaid kasutab olemasolevaid ITA siseveebi andmeid. ITA andmebaasiga ühendumiseks arendas käesoleva lõputöö juhendaja Tarmo Robal veebiteenusena ITA autentimismooduli. Android rakendus kasutab teenust kontrollimaks kas sisse logimisel sisestatud andmed on korrektsed. Turvalisuse aspektis eraldab see rakenduse ja kasutajate autentimise mis vähendab turvariske, kuna rakendus ise ei oma andmeid kasutajate kohta vaid kontroll toimub eraldi seisvas serveris. Samuti on võimalik loodud teenust kasutada ka tulevastes rakendustes mis sisaldavad endas ITA kasutajakontodega sisse logimise komponenti.

Arvutitehnika instituudi raamatukogu platvormi veebiteenus on kasutusel selleks, et oleks võimalik läbi Android rakenduse teostada andmebaasipäringuid. Teenuse loomisel on silmas peetud asjaolu, et seda oleks võimalik kasutada ka teiste rakenduste tarbeks, mis vajavad suhtlemist andmebaasidega, mis asuvad Arvutitehnika instituudi serveris. Päringud, mida teenusega käivitada saab, ei ole otseselt seotud mitte ühegi andmebaasi objektiga, ning neid on võimalik täpselt nii realiseerida, nagu antud olukorras on vaja. See tähendab, et on võimalik suhelda suvalise tabeliga – vastavad väljad peavad sisalduma ainult päringus ning andmebaasis peab eksisteerima antud tabel. Momendil on võimalik teostada ainult *select, insert* ja *update* MySQL operatsioone. Samuti on puuduseks see, et *select* operatsioonil *joini* kasutades, peavad olema liidetavate tabelite kõikide tulpade nimed unikaalsed. Põhjuseks on see, et alati on *select* operatsioonil tagastatavateks väljadeks *, ehk tagastatakse tulemuse kõik väljad. Kui aga liidetavas tabelis on põhitabeliga sama tulp, siis tulemuses kirjutatakse see liidetava tabeli väärtusega üle. Lisaks võib osadel juhtudel takistuseks olla ka päringu *where* osas *or* tingimuse puudumine. Momendil on võimalik kasutada ainult *and* tingimusi. ATI eLibrary rakenduse jaoks veebiteenuse olemas olev funktsionaalsus on piisav, kuid teatud juhtude lahendamiseks vajaks teenus täiendamist.

Android rakendus ATI eLibrary on üks osa raamatukogu platvormist ning funktsioneerib eelkõige haldussüsteemi juhtpuldina. Rakenduse peamine eesmärk on muuta võimalikult mugavaks ja kiireks olemasolevate ja tulevaste raamatute katalogiseerimine. Tänu ribakoodi

lugemise ja Google Books veebiteenuselt raamatu detailse info pärimise funktsioonile on see võimalik. Lisa funktsioonidena on võimalik sirvida olemasolevate raamatute nimekirjas ning neid laenutada. Rakenduse kõige suurem puudus see et, ribakoodi skaneerimise kvaliteet ja tulemus sõltub täielikult telefoni kaamerast, mis võib põhjustada kehvades valgustingimustes vale skaneerimise tulemuse. Lisaks on skaneerimiseks kasutatav rakendus kolmandate osapoolte arendatud ning puudub otsene ligipääs rakenduse parendamiseks. Ühtlasi, arvestades süsteemi senise kasutamise tagasisidet, vajaks rakendus mõningast lisafunktsionaalsust, et parandada kasutusmugavust. Nimel pole võimalik momendil peale raamatu esimest lisamist raamatu detaile enam muuta, mille puudumine võib osutuda väga ebamugavaks. Siiski nõuab rakendus Android nutitelefoni olemasolu ning seetõttu vajaks raamatukogu platvorm ka vähemalt sama funktsionaalsust pakkuvat veebiliidest.

7. Kokkuvõte

Käesoleva tööga loodi Arvutitehnika instituudile raamatukogu haldussüsteem ning sellega täidab püstitatud ülesannet, milleks oli luua Tallinna Tehnikaülikooli Arvutitehnika instituudile kirjanduse baasi paremaks haldamiseks raamatukogu veebiteenuse ja Android rakenduse prototüüp. Töö esimeses pooles anti ülevaade nii veebiteenuste kui ka Android rakenduste loomisest millele lähtuvalt sai loodud raamatukogu platvormi veebiteenus ja mobiilirakendus „ATI eLibrary“.

Loodud veebiteenus on REST arhitektuurile lähtuvalt loodud ning seda on võimalik kasutada ka muudel rakendustel peale „ATI eLibrary“, mis vajavad andmete vahetamist Arvutitehnika instituudi veebiserveris paiknevas MySQL andmebaasiga. Android rakenduse kasutamiseks on vajalik ITA siseveebi kasutajakonto, kuid kuna see on instituudi töötajatel olemas, siis ei nõua rakendus täiendava konto loomist. Peale raamatutebaasi kirjete lisamise, on võimalik rakendust kasutada ka olemasolevate raamatute sirvimiseks ning laenutamiseks. Siiski oleks vajalik nii veebiteenuse kui ka „ATI eLibrary“ mõningased täiendused, et parandada mõlema komponendi funktsionaalsust ja kasutusmugavust. Samuti peaks kuuluma platvormi ka internetibrauserist ligipääsetav veebiliides, mis omaks rakendusega samasugust funktsionaalsust.

Raamatukogu platvorm, mis koosneb veebiteenusest ja Android rakendusest täidab neile seatud eesmärke ning parendab oluliselt Arvutitehnika instituudi kirjanduse haldamist.

Kasutatud kirjandus

[1] Web Services Architecture [WWW]
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#id2260073 / Yves Lafon, (19.04.2015)

[2] Engineering Web Applications: Arhitectural Principles for Web Software/ J. Kuuskeri. Tampere: Tampere University of Technology, 2014

[3] Improving Web Services Security: Scenarios and Implementation Guidance for WCF [WWW]
https://msdn.microsoft.com/en-us/library/ff650794.aspx / J.D. Meier, C. Farre, J. Taylor, P. Bansode, S. Gregersen, M. Sundararajan, R. Boucher (19.05.2015)

[4] Application Fundamentals [WWW]
https://developer.android.com/guide/components/fundamentals.html (25.05.2015)

[5] Processes and Application Life Cycle [WWW]
http://developer.android.com/guide/topics/processes/process-lifecycle.html (25.05.2015)

Lisa 1. Google Books API parameetri selfLink tagastatav vastus


```
{
 "kind": "books#volume",
 "id": "kjiloAEACAAJ",
 "etag": "tb+SuyhpDT4",
 "selfLink": "https://www.googleapis.com/books/v1/volumes/kjiloAEACAAJ",
 "volumeInfo": {
  "title": "A Pointless Book",
  "authors": [
   "Alfie Deyes"
  ],
  "publisher": "Bonnier Books Limited",
  "publishedDate": "2014-09",
  "description": "Alfie Deyes is a prominent YouTuber and his 'Pointless' blog
has amazing figures: 1.6m subscribers, 5m views per month, 1m Twitter followers,
650k Instagram and 480k Facebook followers. With all the humour and quirkiness
of Alfie's 'Pointless' YouTube site, this book is packed with a host of games,
pranks, and jokes.",
  "industryIdentifiers": [
   {
    "type": "ISBN_10",
    "identifier": "1905825900"
   },
   {
    "type": "ISBN_13",
    "identifier": "9781905825905"
   }
  ],
  "readingModes": {
   "text": false,
   "image": false
  },
  "pageCount": 186,
  "printedPageCount": 186,
  "dimensions": {
   "height": "20.80 cm",
   "width": "13.40 cm",
   "thickness": "1.70 cm"
  },
  "printType": "BOOK",
  "categories": [
   "Humor / General"
  ],
  "averageRating": 3.5,
  "ratingsCount": 43,
  "maturityRating": "NOT_MATURE",
  "allowAnonLogging": false,
  "contentVersion": "preview-1.0.0",
  "imageLinks": {
   "smallThumbnail":
"http://books.google.ee/books/content?id=kjiloAEACAAJ&printsec=frontcover&img=1&
zoom=5&imgtk=AFLRE70TWR-5DM4NUXX3IhU2_Fa86OTTEwz63Kl83a2LIC45xFuCKY0rK-
```

```
uWtTKXMd1dnR0Qcj9qeS-NS6LVC_tTu6RggF1Qmr75h8lahqbCnB9x-
7Zs84RTzW6_EeZTphTjvrrLYBOA&source=gbs_api",
   "thumbnail":
"http://books.google.ee/books/content?id=kjiloAEACAAJ&printsec=frontcover&img=1&
zoom=1&imgtk=AFLRE73j7hlnOCwRP3W-
3R8bFJAGGiEbeODZE6xqNkahHZPc3kHdWvN2wCahAHiWHzd8NVx5aiqg3ev1MFeJC3j6Gto1R_enf4lr
rM6p256AEZ_scvb1txiB0GUYrjZYnuow_KMqE3Qu&source=gbs_api"
  },
  "language": "en",
  "previewLink":
"http://books.google.ee/books?id=kjiloAEACAAJ&hl=&source=gbs_api",
  "infoLink": "http://books.google.ee/books?id=kjiloAEACAAJ&hl=&source=gbs_api",
  "canonicalVolumeLink":
"http://books.google.ee/books/about/A_Pointless_Book.html?hl=&id=kjiloAEACAAJ"
 },
 "saleInfo": {
  "country": "EE",
  "saleability": "NOT_FOR_SALE",
  "isEbook": false
 },
 "accessInfo": {
  "country": "EE",
  "viewability": "NO_PAGES",
  "embeddable": false,
  "publicDomain": false,
  "textToSpeechPermission": "ALLOWED",
  "epub": {
   "isAvailable": false
  },
  "pdf": {
   "isAvailable": false
  },
  "webReaderLink":
"http://books.google.ee/books/reader?id=kjiloAEACAAJ&hl=&printsec=frontcover&out
put=reader&source=gbs_api",
  "accessViewStatus": "NONE",
  "quoteSharingAllowed": false
 }
}
```

Lisa 2. Arvutitehnika instituudi raamatukogu platvormi veebiteenuse dokumentatsioon

Päringu formaat:

```
GET: https://www.veebiTeenuseAadress.ee?method=operatsioon

POST: user_id=kasutaja_id&key=unikaalne_võti& request=json_objekt
```

Väljade tüübid ja kirjeldused:

- ***method*** – String, veebiteenuses teostatav operatsioon. Toetatud operatsioonid:
  - *login*
  - *insert*
  - *select*
  - *update*
- ***user_id*** – Integer, veebiteenust kasutava isiku unikaalne identifikaator. Arvutitehnika instituudi Android rakenduse puhul võrdsustatakse ITA kasutajate id-ga.
- ***key*** – String, räsifunktiooniga genereeritud unikaalne võti tagamaks turvalise ühenduse veebiteenusega. Kasutajate lõikes unikaalne.
- ***request*** – String, stringi kujul JSON andmeobjekt. Sisaldab endas veebiteenuses toimepandava operatsiooni detaile. JSON objekti kõik võimalikud väljad:
  - *table* – String, andmebaasi tabel mille poole tahetakse pöörduda
  - *where* – String, JSON andmeobjekt, mis sisaldab endas SQL päringu *where* komponente, *key* positsioonil on tulp ja *value* on väärtus. Andmeobjektil piire ei ole.
  - *group* – String, SQL päringu grupeeringu komponent. Väärtuseks on tulp mille järgi soovitakse grupeerida. Maksimaalselt üks võimalik grupeering.
  - *order* – String, SQL päringu järjestamise komponent. Väärtuseks tulp mille järgi soovitakse tulemusi kasvavalt järjestada. Maksimaalselt üks võimalik väärtus.
  - *join* – Massiiv, massiivi väärtused on JSON andmeobjektid, mis määravad ära SQL lause liitmise komponendid. Üksiku andmeobjekti erinevad väljad:

        - *type* – String, liitmisoperatsiooni tüüp, toetatud kõik MySQL liitmiste tüübid.
        - *table* - String, tabel mida soovitakse põhitabeli(requesti JSON objekti väli table) külge liita.
        - *column1* – String, põhitabeli tulp alusel liidetavat tabelit liidetakse.
        - *column2* – String, liidetava tabeli tulp mille alusel seda liidetakse põhitabeliga
        - *where* – String, JSON andmeobjekt, mis sisaldab endas lisanduvaid kohustuslike(*and*) liitmistingimusi, *key* positsioonil on tulp koos tabeliga(„tabel.tulp") ja *value* on väärtus(juhul kui teise liidetava tabeli tulp, siis „tabel.tulp“). Andmeobjektil piire ei ole.
    - *body* – String, JSON andmeobjekt, mis sisaldab endas uue andmebaasi kirje detaile või muudtava kirje muudetavaid komponente. *Key* positsioonil on tulp ja *value* on väärtus. Maksimaalselt saab korraga määrata ühe kirje detailid.

Kõikide päringute puhul on tagastuseks String, mis sisaldab endas JSON andmeobjekti, millel on üks parameeter:

- ***response*** – String, teavitus, mis saadetakse peale päringu läbiviimist kasutajale, select operatsiooni puhul JSON andmeobjekt.

Lisa 3. Arvutitehnika instituudi raamatukogu veebiteenuse näidispäringud ja vastused

- Päring 1. POST argumendid kui küsitakse kõik raamatud *select* operatsiooniga:
  `user_id=666&key=912ec803b2ce49e4a541068d495ab570&request={"table":"books"}`
- Päring 2. POST argumendid kui küsitakse üks kindel raamat *where* tingimusega *select* operatsiooniga:
  `user_id=666&key=912ec803b2ce49e4a541068d495ab570&request={"table":"books", "where":{"book_id":"3"}}`
- Päring 3. POST argumendid kui küsitakse kõik raamatud *select* operatsiooniga ning tagastav tulemus on grupeeritud välja *isbn* järgi ning järjestatud välja *created_date* alusel:
  `user_id=666&key=912ec803b2ce49e4a541068d495ab570&request={"table":"books", "group":"isbn", "order":"created_date"}`
- Päring 4. POST argumendid kui küsitakse *select* operatsiooniga ühe kindla raamatuga seotud laenud, kasutades *join* parameetrit:
  `user_id=666&key=912ec803b2ce49e4a541068d495ab570&request={"table":"books", "where":{"book_id":"3"}, "join": [{"type":"left", "table":"books_loaned", "column1":"book_id", "column2":"books_id", "where":{"books_loaned.lender_id":"666"}}]}`

  Päring 5. POST argumendid kui lisatakse uus raamat andmebaasi *insert* operatsiooniga:
  `user_id=666&key=912ec803b2ce49e4a541068d495ab570&request={"table":"books", "body":{"title":"Programmeerimine C-keeles", "author":"Viktor Leppikson", "isbn":"9789985850329", "no_of_pages":"192",`

"description":"Raamat on eelkõige mõeldud üliõpilastele ja kõrgkooli lõpetanud insenere-praktikuile. Kuna lugejailt eeldatakse mõningaid programeerimisalaseid kogemusi, näiteks vajaduse korral Pascaliga toimetulekut. Raamat sobib ka neile, kes on otsustanud C-keele iseseisvalt selgeks saada. Eraldi tähelepanu on pööratud algajate poolt tavaliselt tehtavatele vigadele.", "publisher":"Külim"}}

- Päring 6. POST argumendid kui uuendatakse ühe kindla raamatu andmeid *update operatsiooniga*:

user_id=666&key=
912ec803b2ce49e4a541068d495ab570&request={"table":"books", "body":{"image_link":"https://pilt.raamatukoi.ee/pildid/v/0000/110.jpg"}, "where":{"book_id":"5"}}

- Vastus 1. Kuvatav vastus, kui lisatakse uus raamat andmebaasi *insert* operatsiooniga:

{"response":"New record has id: 6"}

- Vastus 2. Kuvatav vastus, kui uuendatakse olemasolevat raamatud *update* operatsiooniga:

{"response":"Updated item in table books"}

- Vastus 3. Kuvatav vastus, kui küsitakse raamatute andmeid *select* operatsiooniga:

{"response":{"count":1,"results":
[{"book_id":"2","title":"Teaching for Quality Learning at University","author":"John Burville Biggs","isbn":"9780335201716","no_of_pages":"250","description":"Since the first edition of Teaching for Quality Learning at University, the tertiary sector has changed dramatically. Individual teachers, as reflective practitioners, still need to make their own decisions about how they are going to get students actively involved in large classes, to teach international students, and to assess in ways that enhance the

quality of learning.","publisher":"Society for Research into Higher Education 1999","image_link":"","owner":"Siavoosh+Payandeh+Azad","location":"ICT-522","created_by":"93","created_date":"2015-04-23 11:00:01","modified_date":"2015-04-23 11:00:01","is_active":"1"}]}}

- Vastus 4. Kuvatav vastus, kui veebiteenuse kasutaja teostab operatsiooni mis pole toetatud või talle lubatud:

  {"response":"This user does not have permissions to use this Web Service"}

Lisa 4. Klassi UtilConnectionManagerRequest meetod makeRequest()

```
/**
   * variables used
   * List<PostElement> POSTargs
   * String GETargs
   * String requestUrl;
   * @return String serviceResponse
   */
  public String makeRequest(){
    String url = requestUrl + GETargs;
    HttpPost post = new HttpPost(url);
    HttpClient client = getNewHttpClient();
    HttpResponse response;
    String serviceResponse = "";
    try {
      List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
      if(this.getPOSTargs().isEmpty()){
        if(client == null){
          client = new DefaultHttpClient();
        }
        HttpGet httpGet = new HttpGet(url);
        response = client.execute(httpGet);
        serviceResponse = EntityUtils.toString(response.getEntity(), "UTF-8");
      }else{
        for(PostElement postElement : this.getPOSTargs()){
          nameValuePairs.add(new BasicNameValuePair(postElement.getName(),
postElement.getValue()));
```

```
        }

        post.setEntity(new UrlEncodedFormEntity(nameValuePairs));

        response = client.execute(post);

        BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));

        String line;

        while ((line = rd.readLine()) != null) {

          if(line.length()>0 && serviceResponse.equals("")){

            serviceResponse = line;

          }

        }

      }

    } catch (IOException e) {

      e.printStackTrace();

    }

    return serviceResponse;

  }
```

Lisa 5. Kasutusel olevate Android versioonide turuosa seisuga 02.11.2015

Allikas: http://developer.android.com/about/dashboards/index.html

Versioon	Koodnimi	API	Turuosa
2.2	Froyo	8	0.2%
2.3.3 – 2.3.7	Gingerbread	10	3.8%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	3.3%
4.1.x	Jelly Bean	16	11.0%
4.2.x		17	13.9%
4.3		18	4.1%
4.4	KitKat	19	37.8%
5.0	Lollipop	21	15.5%
5.1		22	10.1%
6.0	Marshmellow	23	0.3%