TALLINNA TEHNIKA ÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Võrgutarkvara

Androidi rakenduse loomine Apache Cordova platvormil

bakalaureusetöö

Üliõpilane: Hannes Himma
Üliõpilaskood: 050560IAPB
Juhendaja: Jaagup Irve

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

Käesoleva töö eesmärgiks oli praktilise ülesande kujul uurida Androidi rakenduste loomist Cordova platvormil. Antud platvorm võimaldab mobiiltelefoni rakendusi luua pelgalt HTML, CSS ja Javascripti keeltega. Kuna need keeled ei võimalda otsest pöördumist telefoni riistvara (näiteks kaamera) poole, kasutatakse piirangute vältimiseks API pluginaid, mis ühendavad Androidi puhul Javascripti poolse päringu Java poolse vastusega. Nõndaviisi saab näiteks Javascripti käsuga esile kutsuda telefoni kaamera ja pärast pildi tegemist tulemuse Javascripti koodi vastusena tagastada. Juhuks kui mõne rakenduse vajaduse rahuldamiseks sobivat API pluginat pole, siis saab selle ka ise juurde kirjutada. Cordova platvormil on API pluginate loomine, projekti lisamine ja sealt eemaldamine väga mugavalt lahendatud. Cordova iseloomu tõttu saab Androidi rakendusi ka hõlpsalt teistesse operatsioonisüsteemidesse portida. Tuleb vaid operatsioonisüsteemi põhised API pluginad asendada ning rakendus töötab uues operatsioonisüsteemis samalaadselt.

Praktilise ülesande spetsiifilisemaks teemaks valisin Androidi rakenduse pealkirjaga „Kasulik või kahjulik?“, mis abistaks selle kasutajaid erinevate ainete tuvastamisel nende mõju järgi. Mugavamaks infosisestuseks tahtsin kasutada antud alal kõige silmapaistvamat Tesseract OCR mootorit, mis võimaldaks pildilt teksti tuvastada, et selle sisu saaks seejärel analüüsida. Selleks tuli rakendusele juurde teha API plugin, mis võimaldaks Tesseract OCR mootorit kasutada. Kuna teksti sisu analüüsimiseks oli tarvilik ka andmebaasisüsteem, kasutasin selleks WebSQL mootorit, mida Cordova dokumentatsioonis soovitatakse.

Tulemuseks sai rakendus, mis võimaldab väga hea eraldusvõimega piltidelt teksti pea veatult identifitseerida, kuid eraldusvõime langedes tuleb tekstituvastuses üha rohkem vigu esile. Vigade parandamiseks pakub rakendus kasutajale manuaalset teksti redigeerimise võimalust. Viimase sammuna analüüsitakse teksti sisu ja kuvatakse kasutajale andmed leitud ainete kohta.

Abstract

The main goal of this paper was get first hand experience with the capabilities of Apache Cordova platform, which allows the creation of smartphone applications with purely HTML5, CSS and Javascript languages. Official API plugins with different capabilities for accessing hardware resources are easily available for use. Each providing a Javascript interface to access features written in native code. In order to port the application to a different operating system, one only has to port the native code. The inclusion and removal of API plugins can be accomplished with only one command line command, which makes the system quite user friendly.

The practical part of this paper is involved with the creation of an Android application by the name of „Kasulik või kahjulik?“, which helps in identifying different substances one might come across. To increase user friendliness, Tesseract optical character recognition system was included in the project by the means of creating a new API plugin which would allow users to input data for analysis by taking a picture or choosing one from media library. Since the ultimate goal of the application was to identify substances from input text, a WebSQL database system was implemented, to hold substance data.

The finished Android application has excellent optical character recognition capabilities when using a screenshot or a picture with perfect readability. However naturally taken photos can have subpar results. To alleviate the errors made in character recognition, the user is given the opportunity to correct mistakes made by the OCR engine before sending the text for analysis. After analysis, the user is given the list of found substances and their effects with color coding.

Lühendite ja mõistete sõnastik

OCR	***Optical Character Recognition*** Optiline tähemärkide tuvastamine, ehk pildilt teksti tuvastamine.
API plugin	***Application Programming Interface plugin*** Teegikogu, mida saab rakendusele liita lisafunktsionaalsuse tarbeks.
WebSQL	***Web Structured Query Language*** Javascripti põhine andmebaasisüsteem

Sisukord

1. Sissejuhatus

Kuna olen valdava enamuse oma professionaalsest elust töötanud veebiarendajana, siis olen mobiilse kujunduse loomisel erinevate veebilehtede tarbeks hulgaliselt kogemust omandanud. Sellest olenemata polnud ma enne käesoleva töö realiseerimist loonud ühtegi nutiseadme rakendust.

Tänapäeval on iga avaliku veebilehe loomisel esiplaanis selle mobiilne ühilduvus. Veebilehe sisu ja võimalused peavad olema mugavalt ligipääsetavad sõltumata seadmest, millega veebi külastatakse, sest interneti võimekusega nutiseadmete osatähtsuses on märgata järjest kasvavat trendi. See tähendab, et iga uue veebi arendamisel tuleb juurde arendada ka selle kasutajaliidese kohanemise sõltuvalt seadme piirangutele. Näiteks valdaval enamusel nutiseadmetel pole võimalik naturaalselt vallandada linkide ja nuppude *hover* olekut, mis vallandub lauaarvutis, kui hiire kursor on vaadeldava objekti kohal. See tähendab, et *hover* olekul põhinevad menüüd tuleb selliste seadmete jaoks ümber teha, vastasel juhul pole veebilehe sisu ligipääsetav.

Apache Cordova platvormis saab kõigi nutiseadmete operatsioonisüsteemidele luua kasutajaliidese kohaliku keele (*native language)* asemel HTML, CSS ja Javascripti keeltega. See eripära võimaldab mul enda jaoks uues valdkonnas kasutada oma varasemaid veebiarenduse kogemusi, et kanda oma tugevamaid külgi antud valdkonda üle.

Cordova platvormi võimaluste lähemaks uurimiseks lõin Androidi rakenduse pealkirjaga „Kasulik või Kahjulik“, mis aitab selle kasutajal erinevates toodetes esinevaid kasulikke ja kahjulikke aineid nende nimede abil identifitseerida.

Soovisin kasutajatele pakkuda ainete nimede sisestamiseks mitut võimalust, et rakenduse kasutamise kogemus oleks võimalikult mugav. Selle eesmärgi täitmiseks pidin looma ja kasutama API pluginaid ja uurima alternatiivseid võimalusi API pluginate kasutamise asemel.

Kuna API pluginatega seonduv töömaht oli suur, otsustasin rakenduse esmalt Androidile luua. Edasiselt on võimalik rakenduse hõlpsalt ka teiste operatsioonisüsteemidega nutiseadmetesse portida, tuleb vaid API pluginad asendada või need portida.

2. Kasutatud tehnoloogia

Cordova platvormil nutiseadme rakenduse loomiseks tuleb esmalt sisse seada vastav arenduskeskkond. Samu toimetusi saab teha nii Windowsis kui ka Linuxis, mistõttu valisin arenduskeskkonnaks Ubuntu Linuxi, mille panin tööle Vmware virtuaalmasinana. See võimaldas kogu keskkonnast mugavalt koopiaid teha, et vajadusel varasema seisu taastada, juhuks kui keskkond mingil põhjusel rikneb. Seda tuli ka mõnel korral ette, sest Linux oli minu jaoks võrdlemisi uus keskkond. Negatiivse aspektina kaasnes veidi väiksem jõudlus, mis oli tingitud virtuaallahenduse eripärast.

2.1 Apache Cordova

Cordova platvorm sai alguse PhoneGap nimelisest vabatarkvaralisest platvormist, mille lõi 2009 aastal Nitobi nimeline idufirma. Selle eesmärgiks oli luua keskkond, milles saaks mobiilirakenduste arendamisel kasutada põhiliselt vaid veebis levinumaid keeli nagu HTML5, CSS ja Javascript. Juurde realiseeriti ka API pluginate lahendus, mis võimaldas Javascripti koodiga mugavalt mobiiltelefoni platvormi kohaliku keele (*native language*) poole pöörduda.

Adobe ostis 2011. aastal PhoneGap nimelise brändi koos Nitobi idufirmaga ning annetas vabavaralise tuuma Apache Software Foundation kogukonnale. Järgnevalt on PhoneGap ja Cordova arenenud paralleelselt, esimest arendab Adobe ja teist Apache Software Foundation kogukond.

Juhuks kui Cordova platvormil arendatud rakenduses on kogu funktsionaalsus HTML5, CSS ja Javascript keeltega arendatud ja nutiseadme riistvarale päringuid teha ei tule, saab terve rakenduse ilma muudatusteta kõigi enamlevinumate mobiilsete seadmete jaoks mugavalt ühe korraga kompileerida. Seda selletõttu, et HTML5, CSS ja Javascript keeled töötavad kõigis mobiilsetes seadmetes üldjoontes ühesuguselt. Sellest võiks eeldada, et Cordova platvormil on lihtsustatud rakenduste prototüüpimine.

Kui aga sooviks on ka nutiseadme riistvaralist võimekust kasutada või rakendusse näiteks C või Java keelne kood ühildada, siis tuleb luua või kasutada API pluginaid. Cordova platvormil on palju ametlikke API pluginaid, mis pakuvad Javascripti päringutega ligipääsu laiale valikule nutiseadme riistvara funktsionaalsusele. Enamus ametlikke API pluginaid toetavad korraga mitut erinevat nutiseadme operatsioonisüsteemi, mistõttu ühe plugina installeerimisega saab

korraga toe neile kõigile. Peale lihtsustatud prototüüpimise pakub Cordova platvorm ka kompaktset portimise lahendust.

Juhuks kui rakendusse soovitakse integreerida C või C++ keeles kirjutatud koodi, siis saab API plugina tegemise asemel kasutada ka alternatiivse lahendusena Emscripten kompileerijat.

2.2 Emscripten kompileerija

Emscripteni esimeseks stabiilseks väljalaskeks loetakse 2014 aastal välja tulnud lahendust. See kompileerija töötab LLVM kompileerija baasil ning võimaldab bitt koodi failidest Javascripti kompileerida. Teisisõnu kõigepealt kompileeritakse C või C++ rakendus LLVM bitt koodi ja seejärel sealt Javascripti. Antud lahendusega on Javascripti kompileeritud ka näiteks Unreal Engine 3.

Juhuks kui Emscripteniga saab mugavalt C++ keelse koodi Javascriptiks kompileerida, siis saab seda Cordova rakenduses muutumatuna kasutada igal Javascripti toetataval seadmel. See kaotab ära vajaduse C++ keelse koodi kasutamise otstarbel API plugina kirjutamise ja selle portimise vajaduse. Portimise vajaduse ärahoidmiseks uurisin antud kompileerija sobivust Cordova API pluginate asemel kasutamiseks.

2.3 Tesseract OCR

Tesseract OCR loodi aastatel 1985 kuni 1994 Hewlett Packard labs poolt. Seejärel on seda jooksvalt C keelest C++ keelde üle viidud. 2005. aastal määrati sellele vabatarkvara litsents ning 2006. aastast on Google selle arendust rahastanud.

Rakenduse peamiseks eesmärgiks on piltidelt teksti tuvastamine ning seda loetakse omasuguste hulgas laialdaselt parimaks. Kiired testid näidispiltidega GOCR, Ocrad ja Tesseract mootorite kasutamisel illustreerisid selle arvamuse paikapidavust – Tesseract OCR oli vaieldamatult kõige täpsem.

2.4 jQuery ja jQuery Mobile

jQuery on Javascripti teegikogu, mis pakub platvormide vahelist funktsionaalsust väiksema koguse koodiga. See võimaldab hõlpsamini luua interaktiivseid elemente, mis muutuvad ilma terve lehe taaslaadimiseta.

jQuery Mobile on Javascripti teegikogu, mis on optimeeritud nutiseadmetele. See võimaldab arenduses kasutada puutetundliku ekraani erinevaid puudutuste sündmusi (*swipe, tap,* ..), lahendab ära lehekülgede vahelise liikumise ja pakub ka kasutajasõbralikku tüüpkujundust.

jQuery ja jQuery Mobile teegikogud annavad Javascripti põhise kasutajaliidesega Cordova platvormi rakendusele palju arendusmugavust juurde.

2.5 WebSQL andmebaasisüsteem

WebSQL on Javascripti põhine andmebaasisüsteem, mis on toetatud Android, iOS, BlackBerry 10 ja Tizen operatsioonisüsteemides.

Kasutamise alguses loetakse andmebaas mälusse ja kasutamise vajaduse järgselt kustutatakse mälust ära. See tähendab, et SQL päringud on kiired, kuna need tehakse andmebaasile, mis on juba mälus, kuid selletõttu lisavad need ka lisakoormust nutiseadmele, milles on üldjuhul mälunappus sagedane.

Suur osa SQL päringutest tuleb teha sünkroonse lahendusena üksteise järgselt. Näiteks andmebaasi loomine ja selle järgne päring peavad olema õiges järjestuses, et tulemust anda. See tekitas muidu asünkroonse Javascripti kirjutamises keerukamaid seiku.

3. Kasutajaliidese loomine HTML5, CSS ja Javascripti keeltega, kasutades jQuery ja jQuery Mobile raamistikke

Cordova (CLI) käsurida võimaldab Androidil töötava tühja projekti luua hõlpsalt vaid mõne käsuga tekitades kaustastruktuuri koos vajalike failidega, millest saab mugavalt endale sobiva Androidi rakenduse edasi arendada.

Lõin esmalt tühja „hello world!“ stiilis rakenduse Cordova (CLI) käsurea abil. Seejärel sain pelgalt HTML, CSS ja Javascripti keeltega luua peamise kasutajaliidese esmase versiooni. Kuna Cordova platvormil realiseeritud rakenduse selgrooks on Javascripti põhine kasutajaliides, mis vajadusel erinevate API pluginate poole pöördub valitud sündmuste esinemisel ja informatsiooni pärib, tundus mõistlik rakenduses kasutusele võtta ka jQuery ja jQuery Mobile, mis on ühed populaarseimad Javascripti teegikogud, pakkudes rohkem funktsionaalsust lühema koodiga.

jQuery Mobile juures oli atraktiivseks võimekus luua mitut nutiseadme rakenduse lehekülge ühe HTML5 failina. See tähendab, et esmase laadimisega laetakse kõik rakenduse leheküljed korraga alla ning hoitakse neid mälus, sest õigupoolest on tegemist ühe HTML5 failiga. Niimoodi on rakenduse lehekülgede vaheline liikumine sujuvam. Samuti tähendab see seda, et javascriptiga saab ühelt rakenduse leheküljelt teise lehekülje sisu ilma AJAXi päringu tegemiseta pärida ja seda ka muuta, mis teeb sellised päringud kiiremaks.

jQuery Mobile sisaldab lisaks ka tüüpkujundust, millega saab rakendusele kasutaja jaoks kõige tõenäolisemalt juba tuttava väljanägemise anda ning aitab lahendada nutiseadmele omaseid ekraanile vajutamise sündmuseid.

Kogemuse põhjal võib täheldada, et Apache Cordova platvormi on koos jQuery Mobile ja jQuery teegikogudega kiireks (nutiseadmete rakenduste) prototüüpimiseks väga mugav kasutada.

Lisast 1 leiate käesolevalt kirjeldatud lähenemisega HTML5, CSS ja Javascripti koodi ning ekraanipildid, mis on tehtud Android emulaatorist.

4. Kaamera lahendus Cordova platvormil

Kuna soovisin mugavamaks ainete sisestuseks luua süsteemi, mis pildilt aineid välja loeks, siis tuli rakendusele esmalt pildi tegemise ja valimise võimekus liita. Camera nimelise API plugina saab olemasolevale Cordova rakendusele lisada Cordova käsurea (CLI) abil kõigest ühe käsuga (cordova plugin add cordova-plugin-camera).

Seejärel saab API plugina kirjelduse järgi oma Javascripti koodiga pärida kolmest erinevast asukohast pilte – kaks foto albumi asukohta ja kaamera. Androidi eripära tõttu on albumite asukohad mõlema albumi väljakutsumisel samad, mistõttu reaalselt on Androidil vaid kaks valikut.

Camera plugina edastatud vastusena saab pildi küsida kas ühe suure base64 kodeeringus stringina, või läbi faili aadressi (URI). Juhuks kui kasutada base64 kodeeringut, võib nutiseadmetes mälupuudust esineda, mille tulemusel rakendus töö lõpetab. Kogesin seda ka omal käel mõned korrad. Samuti on base64 kodeeringu kasutamine aeglasem. Selle tõttu tuli eelistada faili aadressi põhist edastamist.

Androidi kõige levinumas KitKat (4.4) nimelises versioonis esineb aga Camera API pluginas pildifaili aadressi edastamisel vigu. Probleemidest möödapääsemiseks tuli lisaparameetrina Camera API plugina päringule lisada pildi maksimaalsed mõõtmeid (proportsioone ei rikuta), mille tõttu pilt Camera API plugina vahemälu kausta kopeeritakse ja alles seejärel aadress edastatakse. Camera API plugina vahemälu kaustas asetsevale pildile ligipääsemiseks tuli vaid väljastatud aadressi veidi töödelda. Samuti paranes väiksemate resolutsioonide kasutamise tõttu pildi analüüsimise kiirus Tesseract OCR poolelt. Kõige suurema tõenäosusega aga vähenes ka veidi tekstituvastuse täpsus.

Juhuks kui Camera API pluginast pärida lisaparameetriga *correctOrientation*, et pilt edastataks väljakutsujale õiget pidi, siis ilmnes testitavatel androididel mälupuudusest tingitud rakenduse sulgemist. Sama probleem esines ka *allowEdit,* lisaparameetri kasutamisega.

Camera API plugina korrektseks tööle saamiseks kulus suur osa tervest rakenduse arendamise ajast, sest erinevate võimaluste kasutamisel ilmnes mälupuudust ja vigasid. Niisiis esialgne kogemus ametlike API pluginate kasutamisel polnud väga kasutajasõbralik.

Lisast 1 leiate ülal kirjeldatud lahendusega Javascripti koodi.

5. Pildilt teksti tuvastamine

Juhuks kui pilt on nutiseadme kaamerast või mälust edastatud, siis järgnevalt tuleb sellelt tuvastada tekst, et rakendus saaks jätkata ainete tuvastamisega. Cordova platvormil arendatud rakendusele saab GOCR (https://github.com/antimatter15/gocr.js) või Ocrad (https://github.com/antimatter15/ocrad.js) tarkvara Javascripti kujul mugavalt lisada. Tegemist on Javascripti teekidega, mis on loodud Emscripten kompileerijaga ja mis mõlemad suudavad eraldiseisvalt pildilt teksti tuvastada.

GOCR.js või Ocrad.js Javascripti teekide kasutamine võimaldaks ära hoida portimise vajaduse. Rakendus oleks kasutatav erinevatel operatsioonisüsteemidel ilma lisatööta. Juba esmaste testide tegemisel oli aga näha, et mõlemate lahenduste teksti tuvastustäpsus on Tesseract OCR mootoriga võrreldes märgatavamalt halvem. Selle tõttu otsustasin edasi minna Tesseract OCR mootori integreerimisega „Kasulik või kahjulik“ rakendusse.

Käesoleval hetkel loetakse Tesseract OCR mootorit kõige täpsemaks tekstituvastajaks, mis suudab sisendpildi puhul väljastada seal leiduva teksti. Kuna see on kirjutatud C++ keeles, siis on selle Cordova platvormile tehtud rakendusse ühildamine potentsiaalselt võimalik läbi Emscripteni kompileerija kasutamise, mis C++ allikast Javascripti tulemi looks. Samuti oleks see võimalik läbi API plugina kirjutamise, mis Tesseract OCR mootoriga Javascripti vahendusel suhtleks.

5.1 Tesseract OCR mootori integreerimine Androidi rakendusele kasutades Emscripteni kompilaatorit

Tesseract OCR kasutab Leptonica (www.leptonica.com), libpng12-dev, libjpeg62-dev, libtiff4-dev ja zlib1g-dev teegikogusid. See tähendab, et need tuleb Tesseract OCR rakendusele lisaks Emscripteni abiga LLVM bitt koodi kompileerida, et tulemuse saaks üheks Javascripti failiks kompileerida.

Sain need küll Emscripteni ametliku dokumentatsiooni järgides esimese sammuna LLVM bitt koodi kompileeritud, kuid edasine Javascriptiks kokku kompileerimine osutus kättesaadamatuks.

Nimelt kõigi bitt koodi failide kokku kompileerimisel esines sama nimega objektide mitmekordseid deklaratsioone, mistõttu kompileerimine andis veateate ja lõpetas tegevuse.

Proovisin llvm-nm abiga sama nimega objektide deklaratsioone vältida eemaldades ja kombineerides erinevaid bitt koodi faile. Faile oli aga väga palju ja llvm-nm abil saadud objektide loetelu väga mahukas.

Sain ka Google Groups foorumi vahendusel ühendust ühe Emscripteni loojaga, Alon Zakaiga, kuid sain vaid viiteid olemasolevale dokumentatsioonile, mida järgides küsimusele lahendust ei leidnud.

Kuna antud küsimusega tegelemine võttis äärmiselt suure aja ilma tulemusteta, otsustasin Emscripteniga Tesseract OCR mootori Javascriptiks kompileerimise asemel kasutada API plugina kirjutamise lahendust.

Näib, et Emscripteniga saab hõlpsalt Javascriptiks kompileerida vaid väikeseid projekte. Kui aga tegemist suurema projektiga, milles kasutatakse paljusid teegikogusid, siis on vajalik põhjalik projekti tundmine ja selle mõningane redigeerimine. See teeb Emscripteni kasutamise suuremate kolmanda osapoole kirjutatud ja hallatavate rakenduste kompileerimisel minu seisukohalt liialt ajanõudlikuks.

5.2 Tesseract OCR mootori integreerimine Androidi rakendusele läbi API plugina loomise

Tesseract OCR mootori koos selleks vajalike teegikogudega (Leptonica, libpng12-dev, libjpeg62-dev, libtiff4-dev ja zlib1g-dev) saab Androidi projektile liita Tess-two nimelise vabavaralise Android API'ga, mis on sisuliselt C++ rakenduse port Javasse.

Cordova projektis tuli selle API mugavaks kasutamiseks luua API plugin, mis rakenduse Javascripti ja Tess-two Androidi API (Java) seoksid. Andsin sellele pluginale nimeks TesseractOCRPlugin. Antud plugin koosneb Javascripti koodist, mis on Javascripti ja Java andmete vahendaja rollis ja Java koodist, mis on loodud Tess-Two teegikogu kasutamiseks. API plugina Java koodis tuleb vastuvõetud pildi aadressi abil *Bitmap* pilt avada ja seda õige orientatsiooni info põhjal keerata, et Tesseract OCR mootor teksti tuvastada saaks.

Kuna Tesseract OCR vajab teksti korrektseks tuvastamiseks ka vastava keele informatsiooni sisaldavat andmefaili tessdata kaustas, siis tuli see fail enne TesseractOCRPluginas kasutamist selle Java osale kättesaadaval kujul edastada. Selleks kasutasin vabavaralist Asset2SD API pluginat ja Cordova File API pluginat, et tessdata kataloogi rakenduse failide hulgast välja

kopeerida ja selle aadressi sobival kujul edastada, et keelekataloogile saaks TesseractOCRPlugina Java koodis mugavalt ligi.

Selleks, et Tesseract OCR tuvastustäpsust tõsta kasutasin Tesseract OCR *whitelist* lisaparameetrit, millega seadsin tähemärgid, mida rakendus üritab tuvastada.

Lisast 1 leiate ülal kirjeldatud lähenemisega kirjutatud Javascripti ja Java koodi.

6. Teksti sisu analüüs WebSQL baasi abil

Pärast pildilt teksti tuvastamist pakutakse kasutajale võimalust sisse loetud teksti parandada. Vajadusel saab ka tagasi liikuda ja mõne teise pildi tuvastamiseks valida. Juhuks kui tekstiga ollakse rahul, võidakse edasi navigeerida teksti sisu analüüsi tulemuste leheküljele.

Teksti sisu analüüsimiseks tuli kasutusele võtta andmebaasisüsteem WebSQL, sest see pakkus Javascriptile kõige suuremat vastavasisulist võimekust. Esmases rakenduse realisatsioonis importisin andmebaasi sisu mälusse csv failidest, kuid leidsin, et kiirema tulemuse saab, kui lisada andmebaasi sisu otse Javascripti faili. Mugavuselt olid mõlemad samaväärsed kui Javascripti failis kasutada mitme realiste stringide notatsiooni.

Teksti sisu analüüsimise ettevalmistamiseks eemaldatakse tuvastatud tekstist Javascripti regulaaravaldise asendamise abil kõik tähemärgid peale numbrite, tähtede, sidekriipsu ja tühikute. Seejärel otsitakse neid sõnu andmebaasist.

Juhuks kui andmebaasi otsingu tulemusel identifitseeritakse teksti sisust mõne aine nimetus, siis märgistatakse see värviga, mis oleneb aine mõjust (roheline – hädavajalik, sinine – kasulik, kollane – ohtlik, punane – kahjulik) ja kuvatakse kasutajale selle kohta rohkem informatsiooni.

Lõin andmebaasi kaks tabelit, millest üks tabel sisaldab iga aine kohta unikaalset id’d, ametlikku nime, selle lühikirjeldust, mõju ja aine klassi ning teine tabel aine unikaalset id’d ja aine nimetuse aliast, millise kirjakujuga võib vastavaid aineid leida. See võimaldab tuvastada ka veidi valesti kirjutatud, erinevates käänetes ja pööretes, või võõrkeelseid aine nimetusi ja kuvada nende kohta korrektset nime ja täiendavat infot.

Lisast 1 leiate antud lahendusega Javascripti koodi ning andmebaasi sisu, mida rakenduse töötamisel mälusse WebSQL andmebaasi loetakse.

7. Hinnang tööle

Käesoleva „Kasulik või kahjulik“ Android rakenduse tegemine andis hea ettekujutuse Cordova platvormi tugevatest ja nõrkadest külgedest. Emscripteni kasutamine võimaldas hinnata selle potentsiaali API pluginate alternatiivina. Kuna iga rakendust pole mugav Emscripteni kompilaatori abil projekti liita, uurisin ka API pluginate loomist TesseractOCRPlugin nimelise plugina loomise näitel. Samuti kasutasin olemasolevaid vabatarkvaralisi API pluginaid.

Kokkuvõtlikult sain Cordova platvormil nutiseadmete loomise protsessist hea ülevaate, mis võimaldab mul ka edaspidiselt oma varasemat veebiarenduse kogemust uute nutiseadmete rakenduste loomisel kasutada.

Lisaks õppisin paremini tundma Linuxi ja Vmware süsteemide kasutamist.

Nutitelefonide jõudluse ja kaamera kvaliteedi kasvades saab Tesseract OCR mootorile serveerida parema eraldusvõimega tehtud pilte. Senikaua võib uurida võimalusi fotolt müra vähendamiseks ja eraldusvõime kasvatamiseks. Järgmise sammuna võib ka Androidi funktsionaalsus portida iOS’i, et rakendust saaks ka seal kasutada.

8. Kokkuvõte

Töö peamiseks eesmärgiks oli saada praktiline kogemus Apache Cordova platvormil Android rakenduste loomisest. Spetsiifilisemalt soovisin luua „Kasulik või kahjulik“ nimelise kasutajaid abistava rakenduse, et õppida seeläbi paremini tundma Cordova platvormi API plugina ja rakenduse arendusprotsessi.

Cordova API kasutamine koos jQuery Mobile teegikoguga võimaldab väga kiiret prototüüpimist. Edasine arendusprotsess on aga Androidi Emulaatori aegluse tõttu küllaltki aeganõudev.

„Kasulik või kahjulik?“ nimelise rakenduse teine versioon on küllaltki kiire ja kasutajasõbralik ning annab perfektse eraldusvõimega piltidelt pea alati sajaprotsendilise tuvastustäpsuse. Perfektse pildi saab aga tekstist vaid ekraanipildi funktsionaalsuse abil – naturaalselt tehtud foto on tihtipeale küllaltki halva eraldusvõimega, mistõttu tulemuseks on enamasti üpriski vigane tekst. Samuti võib esile tulla olukordi, kus nutitelefoni kaameraga tehtud pilt salvestatakse vale orientatsiooniga, mis takistab Tesseract OCR mootoril teksti tuvastamast.

Kuna piltide käitlemine ja Tesseract OCR mootori kasutamine nõuavad palju mälu, tuli rakenduses pildi kvaliteeti vähendada, et ära hoida mälu puudusest tingitud rakenduse sulgemist. Niiviisi on rakendus töökindlam, aga teksti tuvastusel esineb rohkem vigu.

Nende probleemide leevendamiseks on realiseeritud tuvastatud teksti parandamise võimalus ja algse teksti käsitsi sisestamise võimalus.

Samuti lisasin Tesseract OCR mootorile lisaparameetrina nimekirja tähemärkidest, mida võidakse tuvastada. Vastasel juhul ilmus piltidele palju erimärke, mis oli tingitud pildi halvast eraldusvõimest.

Failiõigused on Androidi rakenduses küllaltki piiravad, mistõttu failide poole pöördumine võib olla tavapärasest aeglasem ja probleemirohke.

Emscripteni kasutamine on mugav vaid enda kirjutatud, või väiksemate kolmanda osapoole projektide korral. Suuremate projektide kompileerimine on ajaliselt äärmiselt mahukas, kuna nõuab algse koodi head tundmist ja ümbertegemist. Seetõttu on mõistlik neid pigem API plugina abil süsteemi integreerida.

WebSQL on Javascripti jaoks kõige suurema võimekusega lahendus ning selle kasutamine on seetõttu Cordova Apache lahendustes küllaltki asendamatu.

Kasutatud kirjandus

https://github.com/gkcgautam/Asset2SD

https://github.com/rmtheis/tess-two

http://cordova.apache.org

http://kripken.github.io/emscripten-site

https://jquerymobile.com

https://jquery.com

Lisa 1

Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="Content-Security-Policy" content="default-src
        'self' data: gap: https://ssl.gstatic.com 'unsafe-eval';
        style-src 'self' 'unsafe-inline'; media-src *">
        <meta name="format-detection" content="telephone=no">
        <meta name="msapplication-tap-highlight" content="no">
        <meta name="viewport" content="user-scalable=no,
        initial-scale=1, maximum-scale=1, minimum-scale=1,
        width=device-width">
        <link rel="stylesheet" type="text/css" href="css/index.css">
        <title>Kasulik või kahjulik?</title>
        <link rel="stylesheet" type="text/css" href=
        "css/jquery.mobile-1.4.5.min.css">
        <script type="text/javascript" src="js/jquery-2.1.4.min.js"
        ></script>
        <script type="text/javascript" src=
        "js/jquery.mobile-1.4.5.min.js"></script>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </head>
<body>

<!-- Avaleht -->
<div data-role="page" id="avaleht">

    <!-- Päiseriba -->
    <div data-role="header">
        <h1>Kasulik või kahjulik?</h1>
    </div>

    <!-- Sisu -->
    <div data-role="content">
        <p>Tuvasta koostisosad ja nende mõju pildilt:</p>
        <p><a href="#ocr" data-role="button" class="no-0">Vali pilt
        telefonist</a></p>
        <p><a href="#ocr" data-role="button" class="no-1">Tee uus pilt
        </a></p>
        <p>Otsi koostisosasid käsitsi</p>
        <p><a href="#ocr" data-role="button" class="no-2">Otsing</a></p>
        <label>Koostisosad on kirjutatud:</label>
```

```
        <select class="keel">
            <option value="est">Eesti keeles</option>
            <option value="eng">Inglise keeles</option>
            <option value="fin">Soome keeles</option>
        </select>
        <p>Kustuta rakenduse failid SD kaardilt ja lahku. Rakenduse
        järgmisel kasutamiskorral laetakse need uuesti.</p>
        <p><a href="#" data-role="button" class="uninstall" data-theme=
        "a" class="lahku">Kustuta ja lahku</a></p>
    </div>

</div>

<!-- Leht: #ocr -->
<div data-role="page" id="ocr">

    <!-- Päiseriba -->
    <div data-role="header">
        <h1>Täiendamine/parandamine</h1>
    </div>

    <!-- Sisu -->
    <div data-role="content" >
        <div class="loading">Toimub laadimine..</div>
        <div class="manual-search">
            <p>Otsimiseks kirjutage ained allolevasse kasti ja vajutage
            "Tuvasta ained" nupule</p>
        </div>
        <div class="ocr-search">
            <p>Tekst on pildilt sisse loetud ja kuvatakse alljärgnevalt
            </p>
            <p>Soovi korral saate seda parandada.</p>
        </div>
        <a href="#tulemused" data-role="button" class="tulemused-nupp">
        Tuvasta ained</a>
        <textarea name="ocr-tekst" class="ocr-tekst"></textarea>
        <p><a href="#avaleht" data-direction="reverse" data-role=
        "button" data-theme="a" class="tagasi">Tagasi</a></p>
    </div>

</div>

<!-- Leht: #results -->
<div data-role="page" id="tulemused">

    <!-- Päiseriba -->
    <div data-role="header">
        <h1>Ainete loetelu</h1>
```

```
    </div>

    <!-- Sisu -->
    <div data-role="content">
        <div class="allikas"></div>
        <div class="ained"></div>
        <p><a href="#ocr" class="back-ocr tagasi" data-direction=
        "reverse" data-role="button" data-theme="a">Tagasi</a></p>
    </div>

</div>

</body>
</html>
```

Index.css

```
body {
    -webkit-touch-callout: none;                /* prevent callout to
    copy image, etc when tap to hold */
    -webkit-text-size-adjust: none;             /* prevent webkit from
    resizing text to fit */
    -webkit-user-select: text;                  /* prevent copy paste,
    to allow, change 'none' to 'text' */
    }

/*
**  Ainete mõjude värvid
**  Hädavajalik - roheline
**  Kasulik - sinine
**  Ohtlik - kollane
**  Kahjulik - punane
**
*/

.green {
    color: white;
    background: green;
    text-shadow: none;
    padding: 4px;
    border-radius: 6px;
    }

.blue {
    color: white;
    background: blue;
    text-shadow: none;
    padding: 4px;
    border-radius: 6px;
    }
```

```
.yellow {
    color: black;
    background: yellow;
    text-shadow: none;
    padding: 4px;
    border-radius: 6px;
    }

.red {
    color: #FFF;
    background: red;
    text-shadow: none;
    padding: 4px;
    border-radius: 6px;
    }

.moju {
    background-color: #E9E9E9;
    margin-right: -16px;
    margin-left: -16px;
    padding: 18px;
    border-bottom: 1px solid #DDD;
    border-top: 1px solid #DDD;
    text-transform: capitalize;
    margin-bottom: 8px;
    text-align: center;
    }

.klass {
    text-transform: uppercase;
    font-size: 12px;
    margin: 0px;
    border-bottom: 1px solid #DDD;
    padding: 16px;
    padding-left: 0;
    padding-bottom: 0;
    margin-bottom: 4px;
    }

/*
**  Laadimise div konteineri stiil
*/

.loading {
    height: 34px;
    background-image: url('images/ajax-loader.gif');
    background-position: 16px center;
```

```
    background-repeat: no-repeat;
    text-align: center;
    background-color: #000;
    color: #FFF;
    text-shadow: none;
    border-radius: 31px;
    font-size: 21px;
    line-height: 33px;
    padding: 15px;
    padding-left: 40px;
    max-width: 292px;
    margin: 0 auto;
    }

/*
**  Muud stiilid
*/

a.tagasi {
    background-color: #000 !important;
    color: #FFF !important;
    text-shadow: none !important;
    }

a.uninstall {
    background-color: #970000 !important;
    color: #FFF !important;
    text-shadow: none !important;
    }

span.nimi {
    font-weight: bold;
    }

.ained h2, .ained h3 {
    text-transform: capitalize;
    }
```

index.js

```
var app = {
    // Application Constructor
    initialize: function() {
        this.bindEvents();
    },
    // Bind Event Listeners
    //
    // Bind any events that are required on startup. Common events are:
    // 'load', 'deviceready', 'offline', and 'online'.
    bindEvents: function() {
        document.addEventListener('deviceready', this.onDeviceReady,
        false);
    },
    // deviceready Event Handler
    //
    // The scope of 'this' is the event. In order to call the
    'receivedEvent'
    // function, we must explicitly call 'app.receivedEvent(...);'
    onDeviceReady: function() {
        app.receivedEvent('deviceready');
    },
    // Update DOM on a Received Event
    receivedEvent: function(id) {
        //*
        //
        // Peamine javascript
        //
        //
        //*
        // Dokument valmis
        $(document).ready(function() {

        window.name = '';

        // Sule rakendus pärast failipõhise Tesseracti baasi kustutamist
        function quit() {
            window.requestFileSystem(LocalFileSystem.PERSISTENT, 0,
            function(fileSystem){
                fileSystem.root.getDirectory('tesseract', {}, function(
                dirEntry) {
                    dirEntry.removeRecursively(function() {
                        navigator.app.exitApp();
                        }, function(error) {console.log('failed:'+error
                        );});
                    });
                },
                function() {} );
            }
```

```
// Avalehel failipõhise Tesseract baasi kustutamise nupp
$('.uninstall').bind('tap', function() {
    quit();
    });

// Kontroll - kas Tesseract OCR failipõhine baas on juba
kopeeritud
window.requestFileSystem(LocalFileSystem.PERSISTENT, 0,
    function(fileSystem){
        fileSystem.root.getFile(
        "tesseract/tessdata/est.traineddata",
            {create: false},
            function() {
                console.log('Keelefailid juba kopeeritud! Ärme
                üle kirjuta.');
                },
            function() {
                console.log('hakkan kopeerima');

                // Kasutan Cordova Pluginat
                https://github.com/gkcgautam/Asset2SD
                // Kopeerin keelefailid SD kettale
                asset2sd.copyDir({
                    asset_directory: "www/tessdata",
                    destination_directory: "tesseract/tessdata"
                    },
                    function() { console.log('copy fine');  },
                    function() { console.log('copy error'); }
                    );
                }
            );
    },
    function() {});

/*
**  WebSQL Andmebaas
*/

// Andmebaasi päringute vea funktsioon
function err(transaction, error) {
    console.log("Andmebaasi viga: " + error.message);
    }

// Andmebaasi päringute edu funktsioon
function ok(transaction, results) {
    console.log('Päring edukas!');
    }
```

```
// WebSQL andmebaasi importija
function importDatabase(dbContent) {
    mydb = openDatabase("db1", "1.0", "Ained ja nende mõju",
    1024 * 1024);
    // Loon tabelid
    mydb.transaction(function (t) {
        t.executeSql("DROP TABLE IF EXISTS ained");
        t.executeSql("DROP TABLE IF EXISTS aliased");
        }, function() { /*alert('error');*/ }, function() {
            mydb.transaction(function (t) {
                t.executeSql("CREATE TABLE ained (id INTEGER
                PRIMARY KEY ASC, aineID INTEGER, nimi TEXT,
                kirjeldus TEXT, moju TEXT, klass TEXT)");
                t.executeSql("CREATE TABLE aliased (id INTEGER
                PRIMARY KEY ASC, aineID INTEGER, alias TEXT)");
                },
                function() {
                    //alert('viga tabelite loomisel');
                    },
                function() {
                    row = dbContent.trim().split(">");
                    mydb.transaction(function (t) {
                        // reavahe märk >
                        for(i=0;i < row.length; i++) {
                            // veergude vahede märk ;
                            column = row[i].trim().split(';');
                            if(column.length==5) {
                                aineID = parseInt(column[0]);
                                nimi = column[1];
                                kirjeldus = column[2];
                                moju = column[3];
                                klass = column[4];
                                t.executeSql("Insert into ained
                                (aineID, nimi, kirjeldus, moju,
                                klass) values (?,?,?,?,?)", [
                                aineID,nimi,kirjeldus,moju,klass
                                ], ok, err);
                                }
                            else if(column.length==2) {
                                aineID = parseInt(column[0]);
                                alias = column[1];
                                t.executeSql("Insert into
                                aliased (aineID, alias) values
                                (?,?)", [aineID,alias], ok, err);
                                }
                            }
                        },
```

```
                    function() {
                        console.log('viga');
                        },
                    function() { console.log('edu');});
                });
            });
        }

    // Otsin tuvastatud/manuaalselt sisestatud tekstist aineid
    function contentQuery() {
        jQuery('#tulemused .ained').html('');
        jQuery('#tulemused .allikas').html($('textarea.ocr-tekst').
        val());
        prev_moju='';
        prev_klass='';

        // Märgistan tekstist leitud ainete aliased
        function getAlias(transaction, results) {
            for(i = 0; i < results.rows.length; i++) {
                row = results.rows.item(i);
                // Aliaste värvid sõltuvalt mõjust
                switch(row.moju) {
                    case "hädavajalik":
                        color = "green";
                        divClass = "hadavajalik";
                        break;
                    case "tervislik":
                        color = "blue";
                        divClass = "kasulik";
                        break;
                    case "ohtlik":
                        color = "yellow";
                        divClass = "ohtlik";
                        break;
                    case "kahjulik":
                        color = "red";
                        divClass = "kahjulik";
                        break;
                    default:
                        color: "black";
                    }
                // Otsin vaid selliseid aliaseid, mis on eraldi sõnad
                regex = new RegExp("\\b"+row.alias+"\\b",'gi');
                // Märgin ära tekstist leitud ainete aliased

                allikas = jQuery('#tulemused .allikas').html().
                replace(regex,'<span class="'+color+'" >'+row.alias+
                '</span>');
```

```
                // Uuendan tulemuste lehel teksti märgistatud
                aliasega
                jQuery('#tulemused .allikas').html(allikas);

                // Võtan aliastest unikaalsed ID-d
                if(window.name.search(row.nimi+row.aineID)==-1) {
                    if(window.name.length > 0)
                        window.name = window.name+','+row.nimi+row.
                        aineID;
                    else
                        window.name = row.nimi+row.aineID;

                    if((prev_moju=='') || (prev_moju !=row.moju)) {
                        prev_moju = row.moju;
                        jQuery('#tulemused .ained').append('<h4
                        class="'+divClass+' moju"><span class="'+
                        color+'"> '+row.moju+'</span></h4>');
                        prev_klass = row.klass;
                        jQuery('#tulemused .ained').append('<h4
                        class="klass">'+row.klass+'</h4>');
                        }

                    if((prev_klass=='') || (prev_klass !=row.klass))
                     {
                        prev_klass = row.klass;
                        jQuery('#tulemused .ained').append('<h4
                        class="klass">'+row.klass+'</h4>');
                        }

                    jQuery('#tulemused .ained').append('<div
                    class="row"><span class="nimi">' + row.nimi  +
                    '</span> <span class="kirjeldus">' + row.
                    kirjeldus + '</span></div>');
                    }
                }
            }

        mydb.transaction(function (t) {
            // Võtan vaid andmebaasi otsinguks vajaliku osa
            tuvastatud tähemärkidest
            sisu = jQuery('#tulemused .allikas').html().replace(
            /[^0-9a-zA-ZüõöäÖÄÜÕ\- ]/g,' ').replace(/  +/g, ' ').trim
            ().split(' ');
            for(i=0;i<sisu.length;i++) {
                if(sisu[i].length==1) {
                    if(!(typeof sisu[i-1] === 'undefined'))
                        sisu.push(sisu[i-1] + ' ' +sisu[i]);
                    if(!(typeof sisu[i+1] === 'undefined'))
                        sisu.push(sisu[i] + ' ' +sisu[i+1]);
                    }
                }
```

```
            sisu = '"'+ sisu.join('","').toLowerCase() + '"';
            console.log(sisu);
            t.executeSql('SELECT * from ained inner join aliased on
            ained.aineID = aliased.aineID where LOWER(aliased.alias)
            in ('+sisu+') order by CASE ained.moju WHEN "hädavajalik"
            THEN 1 WHEN "tervislik" THEN 2 WHEN "ohtlik" THEN 3 WHEN
            "kahjulik" then 4 ELSE 999 END, ained.klass ASC', [],
            getAlias, err);
            },
            function() { console.log('websql error'); },
            function() { /* success */ });
        }

        // 3. Teen päringu Tesseract OCR-ile, et pildist teksti saada
        function ocrQuery(imageURI) {
            imageURI = imageURI.replace("file://","");
            imageURI = imageURI.split('?')[0];
            // Kasutan Cordova File ja File Transfer pluginaid
            window.requestFileSystem(LocalFileSystem.PERSISTENT, 0,
                function(fileSystem){
                    fileSystemUrl = fileSystem.root.toURL();

                    fileSystemUrl = fileSystemUrl.replace("file://",""
                    );
                    if($("select.keel option:selected").val().length<1
                    )
                        keel = 'est';
                    else
                        keel = $("select.keel option:selected").val();

                    // Kasutan enda tehtud pluginat
                    com.goalbasedsolutions.TesseractOCRPlugin
                    // et kasutada https://github.com/rmtheis/tess-two
                     android toolsi
                    // pildi tuvastamisel
                    TesseractOCRPlugin.ocr(
                        imageURI,
                        fileSystemUrl + "tesseract/",
                        keel,
                        function(recognizedText) {
                            //alert('ocrOutput:'+recognizedText);
                            $.mobile.changePage('#ocr');
                            // Täidan teksti kasti ocr-i tuvastatud
                            tekstiga
                            $('textarea.ocr-tekst').val(recognizedText
                            );
                            // Tuleb ka siia panna, sest alumine
                            .change() funktsioon ei vallandu ilma
```

```
                        vigadeta igal pool
                        $("textarea.ocr-tekst").height($(
                        "textarea")[0].scrollHeight);
                        $('.ocr-search').css('display','block');
                        $('.loading').css('display','none');

                        //alert(imageURI);
                        //alert(imageURI.split('/sdcard/')[1]);
                        delete_file = imageURI.split('/sdcard/'
                        )[1];

                        // Muidu kiilub varsti kinni
                        window.requestFileSystem(LocalFileSystem
                        .PERSISTENT, 0, function(fileSystem){
                            fileSystem.root.getFile(delete_file,
                             {}, function(fileEntry) {
                                fileEntry.remove(function() {
                                    }, function(error) {console.
                                    log('failed:'+error);});
                                });
                            },
                            function() {});
                        },
                    function(error){
                        alert('Pildi lugemise viga: '+error);
                        });
                },
            function(){
                alert("requestFileSystem failed!");
                });
        }

$('textarea.ocr-tekst').change(function() {
    // Suurendan teksti kasti selle sisust sõltuvalt, et ei
    peaks kerima
    $("textarea.ocr-tekst").height($("textarea")[0].scrollHeight
    );
    });

// Pildi tegemine/valimine ei õnnestunud
function getPicFailed(message) {
    alert('Pildi valimise viga: '+message);
    }

function getPic(imgFromWhere) {

    navigator.camera.getPicture(
        ocrQuery,
        getPicFailed,
        {
```

```
            destinationType: Camera.DestinationType.FILE_URI,
            sourceType: imgFromWhere,
            quality : 100,
            targetHeight: 1000,
            targetWidth: 1000,
            encodingType: Camera.EncodingType.JPEG
            });
        }

/*
** Avaleht
*/

// Pildi valimine mälust
$('.no-0').bind( "tap", function() {
    $.mobile.changePage('#ocr');
    $('.manual-search').css('display','none');
    $('.ocr-search').css('display','none');
    $('.loading').css('display','block');
    // PHOTOLIBRARY = 0; (Camera.PictureSourceType)
    getPic(0);
    });

// Pildi tegemine
$('.no-1').bind( "tap", function() {
    $.mobile.changePage('#ocr');
    $('.manual-search').css('display','none');
    $('.ocr-search').css('display','none');
    $('.loading').css('display','block');
    // CAMERA = 1; (Camera.PictureSourceType)
    getPic(1);
    });

// Manuaalne otsing
$('.no-2').bind( "tap", function() {
    $.mobile.changePage('#ocr');
    $('.manual-search').css('display','block');
    $('.ocr-search').css('display','none');
    $('.loading').css('display','none');
    });

/*
** OCR tulemuste leht
*/

// Viib tulemuste lehele ja kutsub esile vajaliku päringu
$('.tulemused-nupp').bind( "tap", function() {
    window.name = '';
    contentQuery();
```

```
        });

        /*
        **  Tulemuste leht
        */

        // Viib tulemuste lehelt tagasi ja puhastab tulemused
        $('.back-ocr').bind( "tap", function() {
            $.mobile.changePage('#ocr');
            $('#tulemused .ained').html('');
            $('#tulemused .allikas').html('');
            });

// Impordin andmebaasi
// Esitan lõputöö dokumendis eraldiseisvalt
importDatabase('\
..\
..\
..');

            }); // End of document ready
        }
    };
app.initialize();
```

TesseractOCRPlugin.js

```
/*
** KASUTAMINE

var success = function(recognizedText) {
    alert(recognizedText);
    }

var failure = function(error) {
    alert("Error: "+error);
    }

tesseractocrplugin.ocr("/mnt/sdcard/images/img01.jpg",
"/mnt/sdcard/tesseract/tessdata/eng.traineddata", "eng", success, failure);
*/
module.exports = {
    ocr: function (image_url, language_url, language_tag, successCallback,
     errorCallback) {
        cordova.exec(successCallback, errorCallback, "TesseractOCRPlugin",
         "ocr", [image_url, language_url, language_tag]);
    }
};
```

TesseractOCRPlugin.java

```
package com.goalbasedsolutions;

import org.apache.cordova.CallbackContext;
import org.apache.cordova.CordovaPlugin;
import org.json.JSONArray;
import org.json.JSONException;
import com.googlecode.tesseract.android.*;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import java.io.IOException;
import android.media.ExifInterface;

public class TesseractOCRPlugin extends CordovaPlugin {

    // Javascripti poolt väljakutsutava tegevuse nimi
    public static final String ACTION_OCR = "ocr";

    // Pildi orientatsiooni väljastaja
    public static int getImageOrientation(String imagePath) {
        ExifInterface exifInterface;
        try {
            exifInterface = new ExifInterface(imagePath);
            }
        catch (IOException err) {
            return ExifInterface.ORIENTATION_UNDEFINED;
            }
        return exifInterface.getAttributeInt(ExifInterface.
        TAG_ORIENTATION, ExifInterface.ORIENTATION_UNDEFINED);
        }

    // Pildi pööraja
    public static Bitmap rotateBitmap(Bitmap bitmap, int orientation) {
        Matrix matrix = new Matrix();
        switch (orientation) {
            case ExifInterface.ORIENTATION_NORMAL:
                return bitmap;
            case ExifInterface.ORIENTATION_FLIP_HORIZONTAL:
                matrix.setScale(-1, 1);
                break;
            case ExifInterface.ORIENTATION_ROTATE_180:
                matrix.setRotate(180);
                break;
            case ExifInterface.ORIENTATION_FLIP_VERTICAL:
                matrix.setRotate(180);
```

```
                matrix.postScale(-1, 1);
                break;
            case ExifInterface.ORIENTATION_TRANSPOSE:
                matrix.setRotate(90);
                matrix.postScale(-1, 1);
                break;
            case ExifInterface.ORIENTATION_ROTATE_90:
                matrix.setRotate(90);
                break;
            case ExifInterface.ORIENTATION_TRANSVERSE:
                matrix.setRotate(-90);
                matrix.postScale(-1, 1);
                break;
            case ExifInterface.ORIENTATION_ROTATE_270:
                matrix.setRotate(-90);
                break;
            default:
                return bitmap;
            }
        try {
            Bitmap bmRotated = Bitmap.createBitmap(bitmap, 0, 0, bitmap.
            getWidth(), bitmap.getHeight(), matrix, true);
            bitmap.recycle();
            return bmRotated;
            }
        catch (OutOfMemoryError e) {
            e.printStackTrace();
            return null;
            }
        }

// Plugina peamine funktsionaalsus
@Override
public boolean execute(String action, JSONArray args,
CallbackContext callbackContext) throws JSONException {
    try {
        if (ACTION_OCR.equals(action)) {
            // Argumentide lugemine
            String language_url = args.getString(1);
            String language_tag = args.getString(2);
            String pathName = args.getString(0);
            pathName = pathName.replace("file://","");

            // Tess-Two libra peamise objekti väljakutsumine
            TessBaseAPI baseApi = new TessBaseAPI();
            baseApi.init(language_url, language_tag);

            // Tuvastame vaid endale huvi pakkuvad või loetavust
            parandavad tähemärgid
```

```
                String whiteList =
                "abcdefghijklmnopqrstuvõäöüxyzwABCDEFGHIJKLMNOPQRSTUVÕÄÖÜ
                XYZW1234567890-,.:'";
                baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST,
                whiteList);

                // Loome Bitmap objekti, et seda õiget pidi keerata ja
                seejärel Tesseract OCR mootorile tuvastamiseks anda
                (Tess-two vahendusel)
                BitmapFactory.Options options = new BitmapFactory.
                Options();
                options.inPreferredConfig = Bitmap.Config.ARGB_8888;
                Bitmap bitmap = BitmapFactory.decodeFile(pathName,
                options);
                Bitmap bmRotated = rotateBitmap(bitmap,
                getImageOrientation(pathName));

                baseApi.setImage(bmRotated);
                String recognizedText = baseApi.getUTF8Text();
                bitmap.recycle();
                bmRotated.recycle();
                baseApi.end();
                // Saadame pildilt tuvastatud teksti tagasi
                callbackContext.success(recognizedText);
                return true;
                }
            callbackContext.error("Invalid action");
            return false;
            }
        catch(Exception e) {
            System.err.println("Exception: " + e.getMessage());
            callbackContext.error(e.getMessage());
            return false;
            }
        }
    }
```

WebSQL andmebaas

```
// Impordin andmebaasi
importDatabase('\
1;A-vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
1;A-vitamiin>\
1;Vitamin A>\
1;Vitamin-A>\
1;Vitamiin A>\
1;Vitamiin-A>\
1;A vitamiin>\
1;beta-carotene>\
1;beta carotene>\
1;beeta-karoteen>\
1;beeta karoteen>\
1;beetakaroteen>\
1;retinol>\
1;retinool>\
2;B1-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
2;B1-Vitamiin>\
2;Vitamiin-B1>\
2;Vitamin B1>\
2;Vitamin-B1>\
2;Vitamiin B1>\
2;B1 Vitamiin>\
2;thiamin>\
2;tiamiin>\
3;B2-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
3;B2-Vitamiin>\
3;Vitamin B2>\
3;Vitamin-B2>\
3;Vitamiin B2>\
3;Vitamiin-B2>\
3;G-vitamiin>\
3;G Vitamiin>\
3;riboflaviin>\
3;B2 Vitamiin>\
3;riboflavin>\
3;vitamiin-G>\
3;vitamin-G>\
3;vitamiin G>\
3;vitamin G>\
4;B3-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
4;B3-Vitamiin>\
4;Vitamin B3>\
4;Vitamin-B3>\
4;Vitamiin B3>\
4;Vitamiin-B3>\
4;B3 Vitamiin>\
4;niacin>\
4;vitamin P>\
4;vitamin PP>\
4;niatsiin>\
4;nikotiinhape>\
```

```
4;nikotiinamiid>\
5;B5-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
5;B5-Vitamiin>\
5;Vitamin B5>\
5;Vitamin-B5>\
5;Vitamiin B5>\
5;Vitamiin-B5>\
5;B5 Vitamiin>\
5;pantothenic acid>\
5;pantoteenhape>\
6;B6-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
6;B6-Vitamiin>\
6;Vitamin B6>\
6;Vitamin-B6>\
6;Vitamiin B6>\
6;Vitamiin-B6>\
6;B6 Vitamiin>\
6;pyridoxine>\
6;pyridoxamine>\
6;pyridoxal>\
7;B7-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
7;B7-Vitamiin>\
7;Vitamin B7>\
7;Vitamin-B7>\
7;Vitamiin B7>\
7;Vitamiin-B7>\
7;B7 Vitamiin>\
7;biotin>\
7;biotiin>\
7;vitamin H>\
7;H-vitamiin>\
7;H vitamiin>\
7;vitamiin H>\
7;vitamiin-H>\
8;B9-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
8;B9-Vitamiin>\
8;Vitamin B9>\
8;Vitamin B9>\
8;Vitamin-B9>\
8;Vitamiin B9>\
8;Vitamiin-B9>\
8;B9 Vitamiin>\
8;folic acid>\
8;folate>\
8;foolhape>\
8;vitamin M>\
8;vitamin-M>\
8;M-vitamiin>\
8;M vitamiin>\
8;vitamiin-M>\
8;vitamiin M>\
9;B12-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
9;B12-Vitamiin>\
9;Vitamin B12>\
```

```
9;Vitamin-B12>\
9;Vitamiin B12>\
9;Vitamiin-B12>\
9;B12 Vitamiin>\
9;cobalamin>\
9;kobalamiin>\
10;C-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
10;C-Vitamiin>\
10;Vitamin C>\
10;Vitamin-C>\
10;Vitamiin C>\
10;Vitamiin-C>\
10;C Vitamiin>\
10;ascorbic acid>\
10;l-askorbiinhape>\
10;l-askorbaat>\
11;D-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
11;D-Vitamiin>\
11;Vitamin D>\
11;Vitamin-D>\
11;Vitamiin-D>\
11;Vitamiin D>\
11;D Vitamiin>\
11;kaltsiferool>\
11;kolekaltsiferool D3>\
11;kolekaltsiferool D2>\
11;ergocalciferol D2>\
11;cholecalciferol D3>\
12;E-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
12;E-Vitamiin>\
12;Vitamin E>\
12;Vitamin-E>\
12;E Vitamiin>\
12;Vitamiin E>\
12;Vitamiin-E>\
12;tocopherol>\
12;tokoferool>\
13;K-Vitamiin;Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;vitamiinid>\
13;K-Vitamiin>\
13;Vitamin K>\
13;Vitamin-K>\
13;Vitamiin-K>\
13;Vitamiin K>\
13;K Vitamiin>\
13;naphthoquinoids>\
13;naftokinoon>\
13;füllokinoon>\
13;menakinoonid>\
13;menakinoon>\
14;kaltsium;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
14;kaltsium>\
14;Calcium>\
14;Ca>\
15:kloriid;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
```

```
15:kloriid>\
15;Chloride>\
15;Cl->\
16;kroom;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
16;kroom>\
16;Chromium>\
16;Cr>\
17;koobalt;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
17;koobalt>\
17;Cobalt>\
17;Co>\
18;vask;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
18;vask>\
18;Copper>\
18;Cu>\
19;Jood;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
19;Jood>\
19;Iodine>\
19;I>\
20:raud;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
20:raud>\
20:Iron>\
20:Fe>\
21;Magneesium;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
21;Magneesium>\
21;Magnesium>\
21;Mg>\
22;Mangaan;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
22;Mangaan>\
22;Manganese>\
22;Mn>\
23;Molübdeen;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
23;Molübdeen>\
23;Molybdenum>\
23;Mo>\
23;Molubdeen>\
24;fosfor;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
24;fosfor>\
24;Phosphorus>\
24;P>\
25;Kaalium;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
25;Kaalium>\
25;Potassium>\
25;K>\
26;Seleen;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
26;Seleen>\
26;Selenium>\
26;Se>\
27;Naatrium;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
27;Naatrium>\
27;Sodium>\
27;Na>\
28;tsink;Mineraali puuduse ärahoidmiseks tuleb iga päev tarbida;hädavajalik;mineraalid>\
28;tsink>\
```

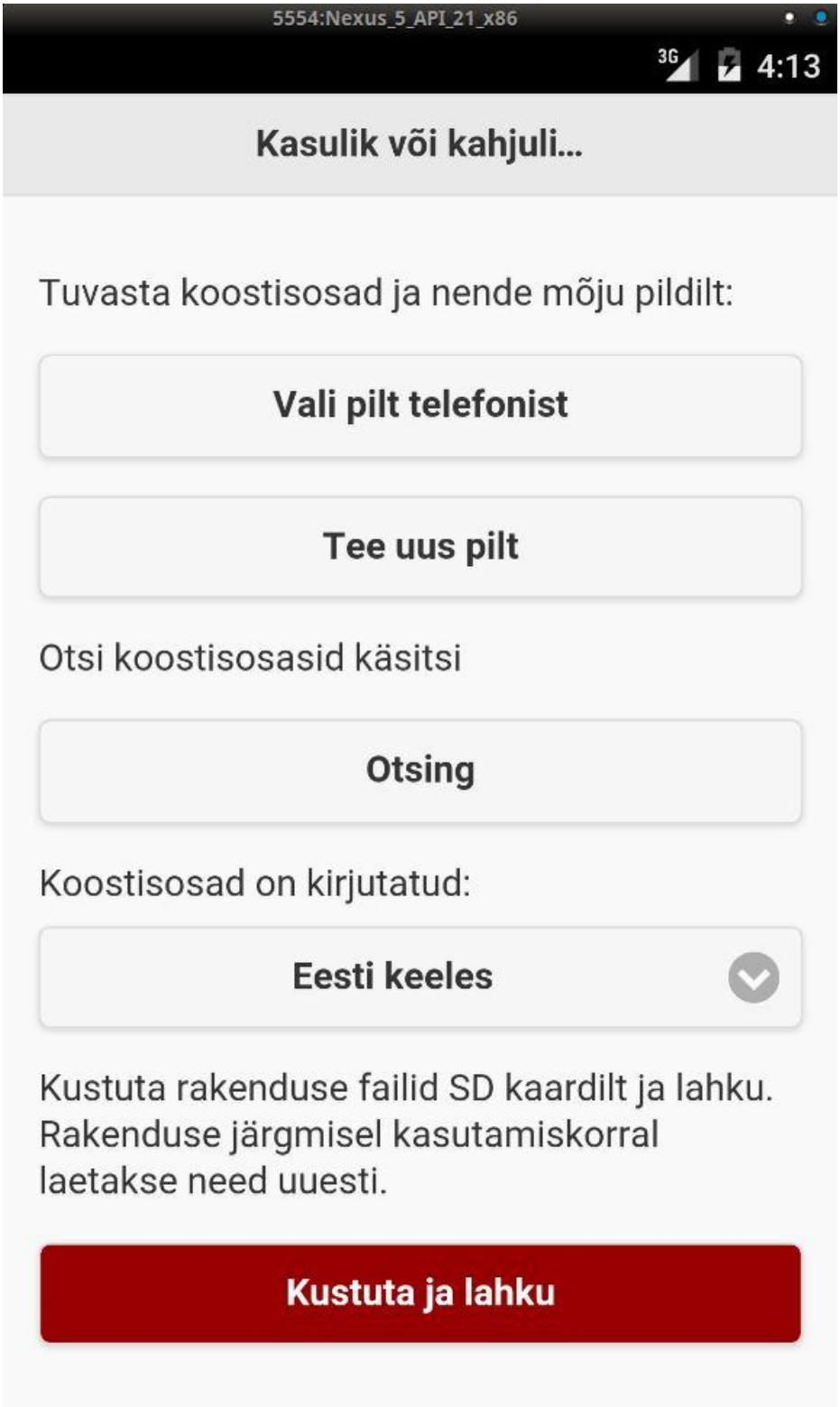
```
28;Zinc>\
28;Zn>\
29;isoleutsiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
29;isoleutsiin>\
29;Isoleucine>\
30;lüsiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
30;lüsiin>\
30;Lysine>\
31;leutsiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
31;leutsiin>\
31;Leucine>\
32;metioniin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
32;metioniin>\
32;Methionine>\
33;tsüstiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
33;tsüstiin>\
33;Cystine>\
33;tsustiin>\
34;Fenüülalaniin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
34;Fenüülalaniin>\
34;Phenylalanine>\
34;Fenuulalaniin>\
35;Treoniin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
35;Treoniin>\
35;Threonine>\
36;Trüptofaan;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
36;Trüptofaan>\
36;Tryptophan>\
37;valiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
37;valiin>\
37;Valine>\
38;histidiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
38;histidiin>\
38;Histidine>\
39;arginiin;Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad
puudulikkuse sümptomid;hädavajalik;aminohapped>\
39;arginiin>\
39;Arginine>\
40;oomega-3;Hädavajalik rasvhape, igapäeva manustamine tugevalt
soovituslik;hädavajalik;rasvhape>\
40;oomega-3>\
40;a-Linolenic acid>\
40;omega-3>\
40;omega 3>\
```

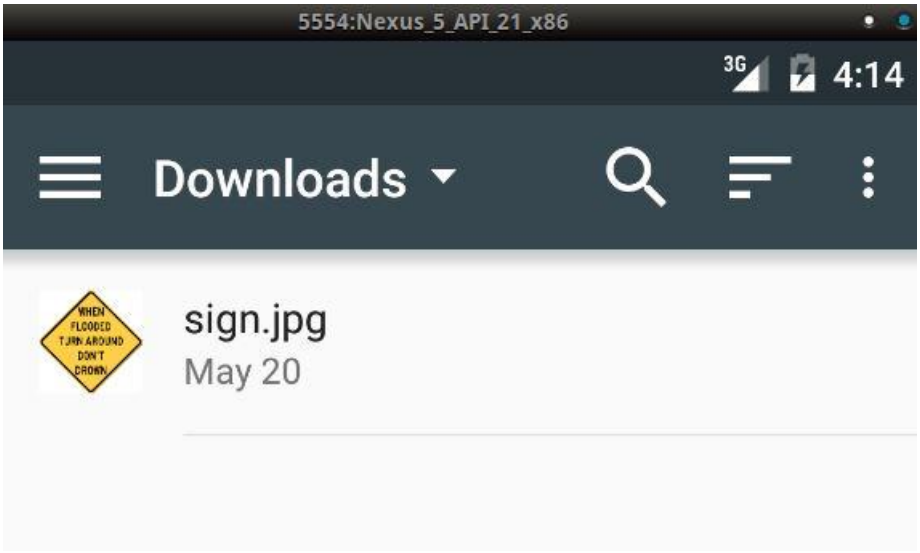
```
40;omega3>\
40;oomega3>\
40;oomega 3>\
40;a-linoleenhape>\
40;alfa-linoleenhape>\
40;ALA>\
40;18:3>\
41;oomega-6;Hädavajalik rasvhape, igapäevane manustamine tugevalt
soovituslik;hädavajalik;rasvhape>\
41;oomega-6>\
41;Linoleic acid>\
41;Linoolhape>\
41;LA>\
41;18:2>\
41;omega-6>\
41;omega 6>\
41;oomega6>\
41;oomega 6>\
42;E102;E-aine;kahjulik;Toiduvärvid>\
43;E104;E-aine;kahjulik;Toiduvärvid>\
44;E110;E-aine;kahjulik;Toiduvärvid>\
45;E122;E-aine;kahjulik;Toiduvärvid>\
46;E123;E-aine;kahjulik;Toiduvärvid>\
47;E124;E-aine;kahjulik;Toiduvärvid>\
48;E127;E-aine;kahjulik;Toiduvärvid>\
49;E210;E-aine;kahjulik;Säilitusained>\
50;E211;E-aine;kahjulik;Säilitusained>\
51;E212;E-aine;kahjulik;Säilitusained>\
52;E213;E-aine;kahjulik;Säilitusained>\
53;E214;E-aine;kahjulik;Säilitusained>\
54;E215;E-aine;kahjulik;Säilitusained>\
55;E218;E-aine;kahjulik;Säilitusained>\
56;E219;E-aine;kahjulik;Säilitusained>\
57;E229;E-aine;kahjulik;Säilitusained>\
58;E221;E-aine;kahjulik;Säilitusained>\
59;E222;E-aine;kahjulik;Säilitusained>\
60;E223;E-aine;kahjulik;Säilitusained>\
61;E224;E-aine;kahjulik;Säilitusained>\
62;E226;E-aine;kahjulik;Säilitusained>\
63;E227;E-aine;kahjulik;Säilitusained>\
64;E228;E-aine;kahjulik;Säilitusained>\
65;E231;E-aine;kahjulik;Säilitusained>\
66;E232;E-aine;kahjulik;Säilitusained>\
67;E239;E-aine;kahjulik;Säilitusained>\
68;E249;E-aine;kahjulik;Säilitusained>\
69;E250;E-aine;kahjulik;Säilitusained>\
70;E251;E-aine;kahjulik;Säilitusained>\
71;E252;E-aine;kahjulik;Säilitusained>\
72;E284;E-aine;kahjulik;Säilitusained>\
73;E285;E-aine;kahjulik;Säilitusained>\
74;E310;E-aine;kahjulik;Antioksüdandid>\
75;E319;E-aine;kahjulik;Antioksüdandid>\
76;E320;E-aine;kahjulik;Antioksüdandid>\
77;E321;E-aine;kahjulik;Antioksüdandid>\
78;E520;E-aine;kahjulik;Mitmesugused lisaained>\
79;E521;E-aine;kahjulik;Mitmesugused lisaained>\
```

```
79;E521;E-aine;kahjulik;Mitmesugused lisaained>\
80;E522;E-aine;kahjulik;Mitmesugused lisaained>\
81;E523;E-aine;kahjulik;Mitmesugused lisaained>\
82;E541;E-aine;kahjulik;Mitmesugused lisaained>\
83;E554;E-aine;kahjulik;Mitmesugused lisaained>\
84;E555;E-aine;kahjulik;Mitmesugused lisaained>\
85;E556;E-aine;kahjulik;Mitmesugused lisaained>\
86;E559;E-aine;kahjulik;Mitmesugused lisaained>\
87;E620;E-aine;kahjulik;Maitsetugevdaja>\
88;E621;E-aine;kahjulik;Maitsetugevdaja>\
89;E622;E-aine;kahjulik;Maitsetugevdaja>\
90;E623;E-aine;kahjulik;Maitsetugevdaja>\
91;E625;E-aine;kahjulik;Maitsetugevdaja>\
92;E905;E-aine;kahjulik;Glaseerained>\
93;E914;E-aine;kahjulik;Glaseerained>\
94;E943a;E-aine;kahjulik;Propellandid>\
95;E943b;E-aine;kahjulik;Propellandid>\
96;E944;E-aine;kahjulik;Propellandid>\
97;E950;E-aine;kahjulik;Kunstlikud magusained>\
98;E951;E-aine;kahjulik;Kunstlikud magusained>\
99;E952;E-aine;kahjulik;Kunstlikud magusained>\
100;E954;E-aine;kahjulik;Kunstlikud magusained>\
101;E962;E-aine;kahjulik;Kunstlikud magusained>\
102;E1201;E-aine;kahjulik;Mitmesugused lisaained>\
103;E1452;E-aine;kahjulik;Mitmesugused lisaained>\
104;E1520;E-aine;kahjulik;Mitmesugused lisaained>\
42;E102>\
43;E104>\
44;E110>\
45;E122>\
46;E123>\
47;E124>\
48;E127>\
49;E210>\
50;E211>\
51;E212>\
52;E213>\
53;E214>\
54;E215>\
55;E218>\
56;E219>\
57;E229>\
58;E221>\
59;E222>\
60;E223>\
61;E224>\
62;E226>\
63;E227>\
64;E228>\
65;E231>\
66;E232>\
67;E239>\
68;E249>\
69;E250>\
70;E251>\
```

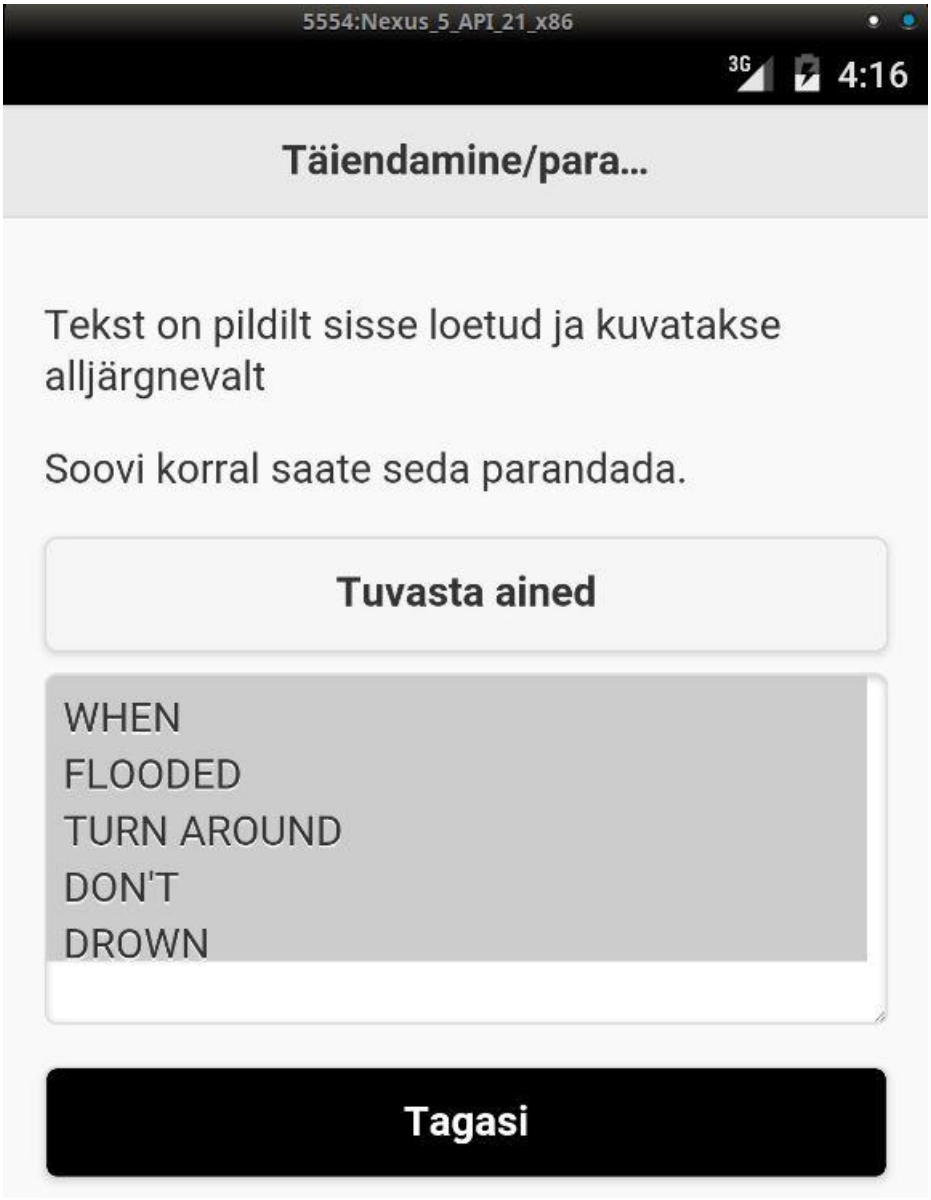
```
71;E252>\
72;E284>\
73;E285>\
74;E310>\
75;E319>\
76;E320>\
77;E321>\
78;E520>\
79;E521>\
80;E522>\
81;E523>\
82;E541>\
83;E554>\
84;E555>\
85;E556>\
86;E559>\
87;E620>\
88;E621>\
89;E622>\
90;E623>\
91;E625>\
92;E905>\
93;E914>\
94;E943a>\
95;E943b>\
96;E944>\
97;E950>\
98;E951>\
99;E952>\
100;E954>\
101;E962>\
102;E1201>\
103;E1452>\
104;E1520>');
```

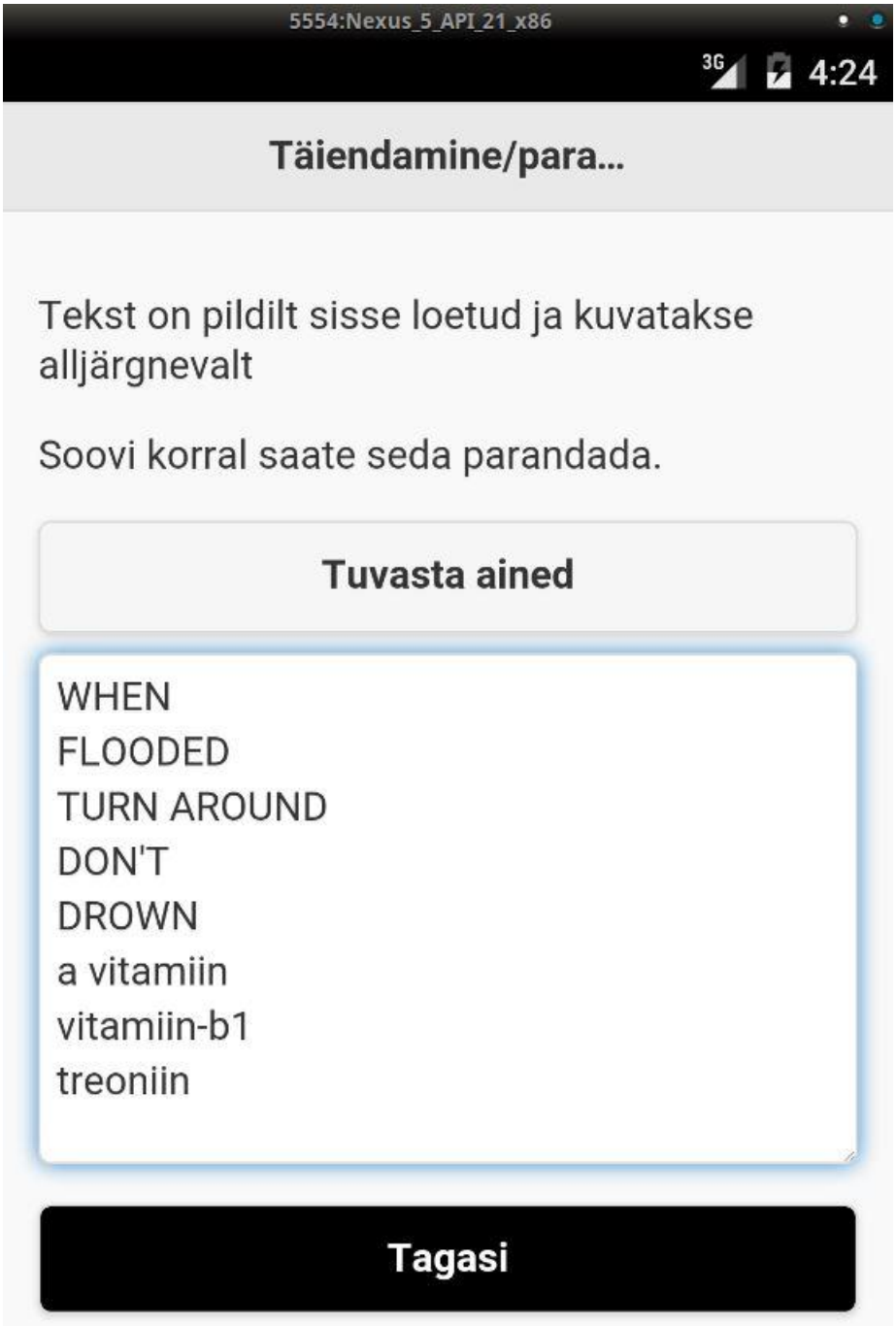
Ekraanipildid rakendusest (emulaator)

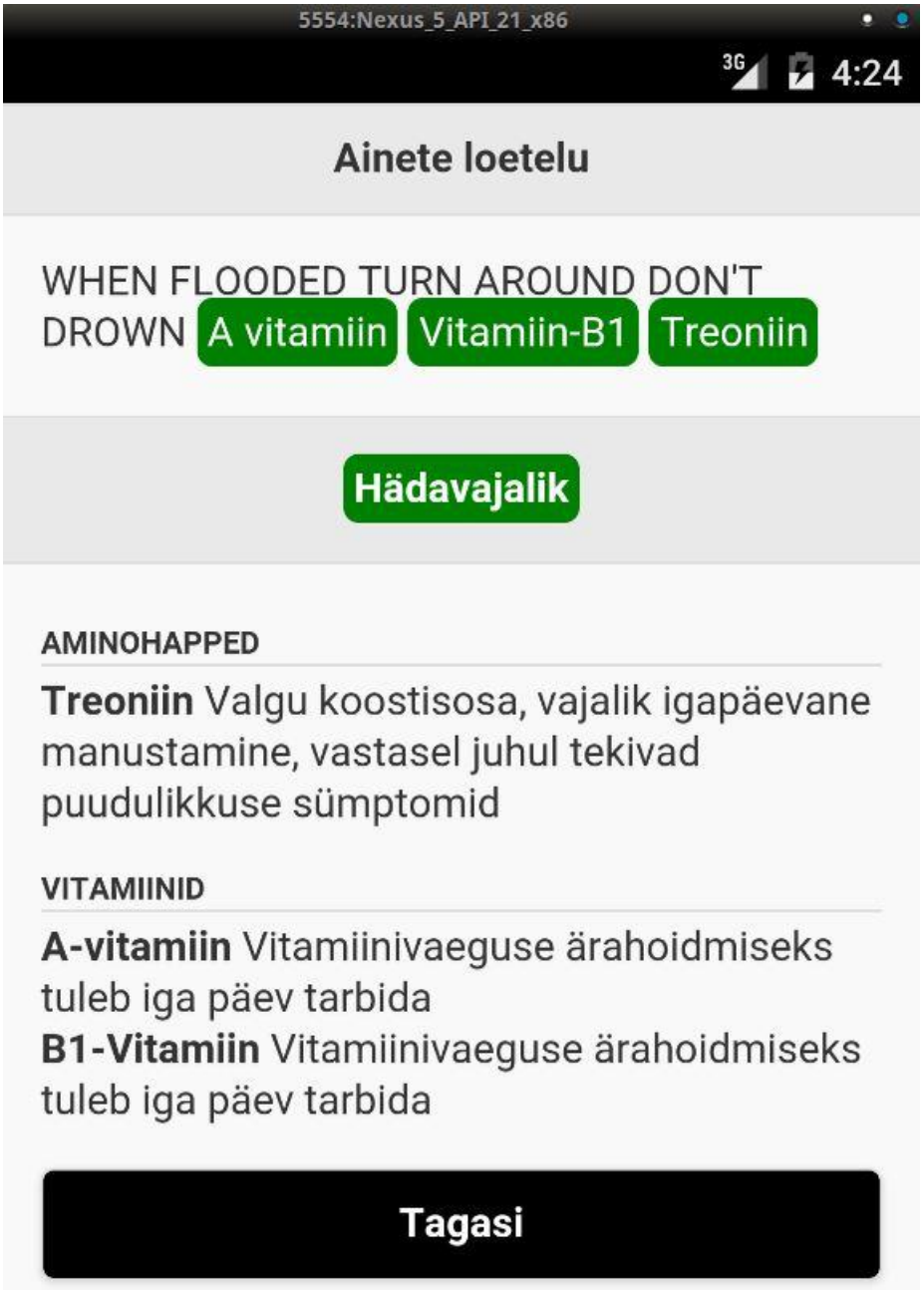



5554:Nexus_5_API_21_x86
4:14
Downloads
WHEN FLOODED TURN AROUND DON'T DROWN
sign.jpg
May 20


5554:Nexus_5_API_21_x86
4:15
Täiendamine/para...
Toimub laadimine..
Tuvasta ained
Tagasi


5554:Nexus_5_API_21_x86
3G
4:16
Täiendamine/para...
Tekst on pildilt sisse loetud ja kuvatakse alljärgnevalt
Soovi korral saate seda parandada.
Tuvasta ained
WHEN
FLOODED
TURN AROUND
DON'T
DROWN
Tagasi


5554:Nexus_5_API_21_x86
3G
4:24
Täiendamine/para...
Tekst on pildilt sisse loetud ja kuvatakse alljärgnevalt
Soovi korral saate seda parandada.
Tuvasta ained
WHEN
FLOODED
TURN AROUND
DON'T
DROWN
a vitamiin
vitamiin-b1
treoniin
Tagasi


5554:Nexus_5_API_21_x86
3G
4:24
Ainete loetelu
WHEN FLOODED TURN AROUND DON'T DROWN
A vitamiin
Vitamiin-B1
Treoniin
Hädavajalik
AMINOHAPPED
Treoniin Valgu koostisosa, vajalik igapäevane manustamine, vastasel juhul tekivad puudulikkuse sümptomid
VITAMIINID
A-vitamiin Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida
B1-Vitamiin Vitamiinivaeguse ärahoidmiseks tuleb iga päev tarbida
Tagasi