TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Marion Laur**

# Mobiilirakendus kuulutuses oleva info automaatseks kalendrisse sisestamiseks

**Bakalaureusetöö (9 EAP)**

Juhendaja: Sven Aller

Tartu 2020

## Mobiilirakendus kuulutuses oleva info automaatseks kalendrisse sisestamiseks

**Lühikokkuvõte:**

Töös vaadeldakse optilise märgituvastuse ja loomuliku keele töötluse vahendite kasutusvõimalusi mobiilirakenduse loomiseks, mis võimaldab automaatselt tuvastada kuulutuselt olulist infot. Samuti hinnatakse protsessi automatiseerimise ajalist kasutegurit võrreldes kalendrisündmuse lisamisega käsitsi meetodil.

**Võtmesõnad:**

Mobiilirakendus, optiline märgituvastus, loomuliku keele töötlus, Android

**CERCS:** P175 (Informaatika, süsteemiteooria)

## Mobile application for automatically creating a calendar event from an event poster

**Abstract:**

This thesis explores the usage of optical character recognition and natural language processing tools within the context of building a mobile application that automatically extracts relevant information from an event poster. A time-efficiency analysis is given comparing the automated process with the manual method of creating a calendar event.

**Keywords:**

Mobile application, optical character recognition, natural language processing, Android

**CERCS:** P175 (Informatics, systems theory)

# Sisukord

# Sissejuhatus

Nutitelefonide kasutamine on suurenenud aasta aastalt ning sellega seoses kiirelt arenev tehnoloogia pakub palju võimalusi inimeste elu lihtsamaks tegemiseks [1]. Selleks on loodud erinevaid mobiilirakendusi, mis hõlbustavad inimestel näiteks pangaga suhtlust, navigeerimist ning toidu tellimist. Mida aeg edasi, seda enam otsivad ka nutitelefonide tootjad viise, kuidas suurendada seadmete jõudlust ja kasutusmugavust [2]. Edasiminekud selles vallas võimaldavad kasutusele võtta uusi tehnoloogiaid, mida varasemalt riistvaralistest ja tarkvaralistest piirangutest tulenevalt nii edukalt rakendada ei olnud võimalik. Ühena nendest võib nimetada optilist märgituvastust.

Üks nutitelefonide kasutamise miinuseid on tekstilise info käsitsi sisestamise aeglus, sest puuteekraanil oleva klaviatuuriga on vigu kerge teha [3]. Sama probleem esineb ka juhul, kui eesmärk on välisest allikast teksti kopeerimine mobiilirakendusse – tegevus on aeganõudev ning tulemus võib olla vigane tulenevalt inimese tähelepanematusest. Näiteks leidub teadetetahvlitel sageli kontserdi-, teatri- vm ürituste kuulutusi, millel olevat infot sooviks kalendris säilitada. Kuna info käsitsi sisestamine on ajakulukas, oleks mõistlikum kuulutust pildistada ning saadud fotolt vajalikku infot automaatselt tuvastada.

Töö eesmärk on valida sobiv optilise märgituvastuse mootor, luua rakendus Android seadmele, mis võimaldab üritust reklaamivast kuulutusest nutitelefoniga pilti teha, sealt automaatselt tuvastada olulist informatsiooni (ürituse pealkiri ning toimumise aeg ja koht) ning pakkuda võimalust leitud andmed ühe nupuvajutusega kalendrisse sisestada. Erinevalt üldotstarbelistest OCR-rakendustest, kus kasutaja peab talle huvipakkuva osa pildist käsitsi märkima, saab töö käigus loodud rakendus sellega iseseisvalt hakkama ning paigutab info ka sobivatesse kohtadesse kalendris.

Bakalaureusetöö on jaotatud kolmeks suuremaks peatükiks. Esimeses peatükis antakse ülevaade nutitelefoniga optilise märgituvastuse tegemise eripäradest, loomuliku keele töötlusest ning olemasolevatest tööriistadest ja rakendustest. Teises peatükis kirjeldatakse valminud lahenduse arhitektuuri ja kasutatud tehnoloogiaid. Kolmandas peatükis esitatakse mobiilirakenduse testimise tulemused ning antakse hinnang valminud tööle tervikuna.

# 1. Taust

Selles peatükis kirjeldatakse optilise märgituvastuse arengut mobiiliseadmetele ning antakse ülevaade, milliseid olemasolevaid optilise märgituvastuse mootoreid on olemas nutitelefonidega kasutamiseks. Samuti antakse ülevaade loomuliku keele töötluse alustaladest ning tutvustatakse vahendeid, mis lihtsustavad tekstist olulise teabe automaatset kättesaamist. Peatüki lõpus tuuakse välja mõned olemasolevad mobiilirakendused kuulutuses oleva info automaatseks kalendrisse sisestamiseks.

## 1.1 Optiline märgituvastus nutitelefoniga

Optiline märgituvastus ehk OCR (*Optical Character Recognition*) kujutab endast käsitsi kirjutatud või väljatrükitud teksti teisendamist masinloetavale kujule. Ajalooliselt kerkis optilise märgituvastuse idee esmakordselt esile seoses võrkkesta skanneri leiutamisega ning aastal 1929 patenteeris Austriast pärit Gustav Tauschek esimese OCR-lugemismasina, mis kasutas valgussensoreid märkide tuvastamiseks [4]. Esimene kommertslik optiline märgituvastussüsteem tekkis 1950ndatel Ameerika Ühendriikides ning iga järgneva aastakümnega on tehnoloogia selles vallas hüppeliselt edasi arenenud ja leidnud aina laiemat kasutust erinevates eluvaldkondades [5].

A. Verma jt on oma artiklis [6] välja toonud, et varasemalt oli optiline märgituvastus suuresti limiteeritud, sest tekstituvastustarkvara jaoks sobilike piltide saamiseks tuli kasutada skannereid, mis olid aeglased ning mille transport oli keeruline. Seetõttu oli ka optilise märgituvastuse põhiliseks rakendusalaks dokumentide digitaliseerimine, mis oma olemuselt ei nõudnud suurt mobiilsust. Samas kirjutavad nad, et tänapäeva kontekstis on erinevate nutiseadmete turule tulek andnud ka optilisele märgituvastusele uue suuna. Näiteks Google'i mobiilirakendus Google Translate võimaldab kasutajal nutitelefoni kaamera abil reaalajas tõlkida 88 keelest rohkem kui 100 toetatud keelde, kasutades optilise märgituvastuse ja liitreaalsuse vahendeid [7].

Erinevalt aga näiteks tekstituvastuse rakendamisest skaneeritud dokumentidele, kus üldjuhul on tegu võrdlemisi kindla formaadiga (teksti ühtlane orientatsioon), on suvaliselt nutitelefoni kaameraga tehtud pildilt märgituvastus oluliselt keerulisem. Raskeks muudab ülesande piltide ettearvamatu sisu: taust võib olla kirju, tekst võib olla eri toonides, kaldus ja ebahariliku nurga all, pilt ise võib olla tehtud kehvalt valgustatud oludes ning mõni oluline osa võib jääda fookusest välja [6]. Samuti võrreldes lauaarvutile suunatud OCR-

süsteemidega peab nutitelefonis arvestama tunduvalt väiksema mälumahu, jõudluse ning muude riist- ja tarkvaraliste eripäradega, mistõttu enamik lauaarvutitele suunatud märgituvastusmootoreid enam mobiilirakenduse kontekstis nii hästi ei päde [8].

## 1.2 Mobiiltelefoni operatsioonisüsteemidele mõeldud optilise märgituvastuse töövahendid

Järgnevalt on nimetatud mõningad populaarsemad märgituvastusmootorid, mis on mõeldud erinevate nutitelefoni operatsioonisüsteemidega kasutamiseks.

- ABBYY Mobile Capture [9] – tasuline arendustarkvara (ingl *software development kit*) Android ja iOS operatsioonisüsteemidele, mis võimaldab automaatset tekstituvastust reaalajas seadmes nii lokaalselt kui ka serveripõhiselt. Lisaks on olemas sisseehitatud funktsionaalsus visiitkaartidelt, passidelt, juhilubadelt ja pangakaartidelt info lugemiseks ning muudelt dokumentidelt spetsiifiliste väljade leidmiseks. Toetatakse 63 keelt.
- Tesseract OCR [10] – põhiline mootor ise küll ei ole otseselt mobiiliseadmetele suunatud, ent dokumenteeritud on lai hulk saadaval olevaid teeke, mis pordivad olulised funktsionaalsused Android ja iOS operatsioonisüsteemile üle. Tesseract OCR on avatud lähtekoodiga ning pakub seega ulatuslikult võimalusi optilist märgituvastust ülesandele vastavalt kohandada, näiteks on võimalik olemasolevaid mudeleid treenida enda spetsiifilise andmestiku peal täpsemaks. Hetkel toetatakse rohkem kui 100 keelt, sh eesti keelt.
- Google Mobile Vision API [11] – rakendusliideste kogumik, mis muuhulgas pakub optilise tekstituvastuse funktsionaalsust (samuti on võimalik tuvastada nägusid ja triipkoode). Võimaldab tuvastada igasugust teksti ladina kirjas. Märgituvastus töötab reaalajas seadmes lokaalselt. Mobile Vision API on migreerumas Firebase'i ML Kit arendustarkvarasse [12].

Tasulisi ja vabavaralisi töövahendeid leidub veelgi, kuid eelpool toodud kolm näidet annavad hea ülevaate praegu saadaval olevate märgituvastusmootorite pakutavatest võimalustest ja omadustest.

## 1.3 Automaatne infootsing tekstist

Loomuliku keele töötlus arvuti abil ja selle kasutusalad on olnud teadlaste huviorbiidis juba eelmise sajandi keskpaigast [13]. Loomulik keel kui inimestevaheline suhtlusviis võib esineda nii pildis, kõnes kui ka kirjas. Nimetatud meediumite süstemaatilise analüüsimise tulemusena on võimalik erinevaid meetodeid rakendades luua masintõlkeprogramme, kõnetuvastussüsteeme, juturoboteid, aga ka näiteks genereerida sisukokkuvõtteid ja klassifitseerida infot.

Tulenevalt loomuliku keele omadustest võib automaatne keeletöötlus osutuda küllaltki keeruliseks ülesandeks. Loomuliku keele töötluse tehnoloogilisest arengust annab hea ülevaate P. M. Nadkarni jt kirjutatud artikkel [14], mis on järgnevalt lühidalt kokku võetud. Kuna fraasi tähendus võib oleneda ümbritsevast kontekstist, siis selle adekvaatseks analüüsiks peab arvuti tekstist ka sisuliselt aru saama, st semantikat mõistma. Ajalooliselt on püütud seda ülesannet lahendada mitmel moel. Üsna ruttu jõuti arusaamale, et keele semantika defineerimine üksnes käsitsi kirjutatud reeglite abil ei tööta eriti hästi, sest selliseid reegleid on keeruline hallata, erijuhte on väga suur hulk ning ootamatud konfliktid on kerged tekkima. Paremaid tulemusi hakati saavutama siis, kui võeti kasutusele tõenäosuslikud meetodid, mis modelleeriti suurte tekstikorpuste põhjal. Statistiline lähenemine võimaldas luua töökindlamaid süsteeme ning ka tänapäeval on kõige paremaid tulemusi saavutatud just statistiliste masinõppemeetodite abil.

Automaatsel tekstianalüüsil on mitmeid praktilisi väljundeid, millest üks on nimeolemite tuvastamine etteantud tekstist. Nimeolemiteks kutsutakse tekstis leiduvaid isikute, organisatsioonide ja geograafiliste asukohtade nimetusi, aga mõnel juhul ka seal esinevaid ajaväljendeid, protsentarve jms [15]. Edukas nimeolemite tuvastus on tihti aluseks keerulisemate ülesannete lahendamisel, näiteks infootsingu tarkvara või küsimustele automaatselt vastava süsteemi loomisel [16]. Nagu loomuliku keele töötluse puhul üldiselt, on ka nimeolemite tuvastamisel erinevaid nüansse. Keeruliseks muudavad selle ülesande näiteks keeles esinevad sünonüümid, homonüümid, sõna tüve muutused, ühe nimeolemi osade paiknemine lauses üksteisest eraldi või nende paiknemine juhuslikus järjekorras; ajaväljendite puhul võib raskeks osutuda ka ajalise järgnevuse määramine, tuleviku ja mineviku sündmuste tuvastamine jne [14].

Automaatseks info leidmiseks tekstist on loodud mitmeid loomuliku keele töötluse vahendeid. Sobiva töövahendi valimisel tuleb arvestada mitme aspektiga. Kindlasti on

oluline, et valitud vahend saaks hakkama konkreetses keeles oleva teksti töötlemisega ning oleks piisavalt täpne. Näiteks meetod, mis on spetsialiseerunud inglise keele analüüsimisele, ei sobi kasutamiseks eesti keelega tulenevalt nende kahe keele struktuurilistest erinevustest. Kui tegu on programmeeritava mooduliga, tuleb arvestada ka programmeerimiskeele sobivusega soovitud kasutusalaga. Veebiteenuseid kasutades on vajalik internetiühenduse olemasolu. Veebipõhised rakendusliidesed pakuvad mõnikord mugavamat integreerimise võimalust väliste tarkvaralahendustega, sest näiteks üle HTTP protokolli suheldes ei sõltu kliendikood serveri poolel kasutusel olevatest teekidest ja programmeerimiskeeltest ning on sellest tulenevalt kergemini hallatav. Samuti suuremamahuliste tekstide puhul võib tekstitöötlus kujuneda piisavalt aeganõudvaks, mistõttu võib efektiivsemaks osutuda pilveteenuse kasutamine. Erinevat sorti töövahendeid on nii vabavaralisi ja avatud lähtekoodiga kui ka tasulisi või selliseid, mis piiravad tasuta päringute arvu või mahtu.

## 1.4 Loomuliku keele töötluse vahendid

Järgnevalt on toodud mõned näited loomuliku keele töötluse vahenditest, mida on võimalik kasutada teksti lingvistiliseks analüüsiks.

- EstNLTK [17] – avatud lähtekoodiga Pythoni teek, mis on mõeldud spetsiaalselt eesti keele töötlemiseks. Võimaldab teostada teksti segmenteerimist, morfoloogilist analüüsi, sünteesi ja grammatika kontrolli ning osalausete, nimeolemite ja ajaväljendite tuvastust. Teegil on olemas ka Eesti Wordneti liides jpm.
- TextBlob [18] – avatud lähtekoodiga Pythoni teek, mis lisaks üldisele morfoloogilisele analüüsile võimaldab ka parandada kirjavigu ning tekste tõlkida. Mõeldud eelkõige inglise keele töötluseks, aga lisadena on olemas ka prantsuse ja saksa keele tugi.
- spaCy [19] – avatud lähtekoodiga Pythoni teek, mis rõhub keeletöötluse kiirusele ja täpsusele. Toetab teksti segmenteerimist rohkem kui 50 keeles ning põhjalikumaks analüüsiks on saadaval erinevaid keelemudeleid 11 keelele (töö kirjutamise hetkel eesti keelele veel mitte). Võimaldab tuvastada suurt hulka nimeolemeid ning neid visualiseerida. Pakutavat funktsionaalsust saab mugavalt täiendada masinõppe meetodite abil.

- Stanford CoreNLP [20] – avatud lähtekoodiga tarkvara, mis on kirjutatud programmeerimiskeeles Java. Olemasolevale rakendusliidesele on saadaval ka hulgaliselt mähkureid teiste programmeerimiskeeltega kasutamiseks. Ametlikult toetab inglise, prantsuse, hispaania, saksa, araabia ja hiina keelt. Teostab üldist morfoloogilist analüüsi, aga täpsemad töötluse võimalused varieeruvad keeleti. Näiteks meelsusanalüüsi on võimalik teha ainult inglise keelega ning nimeolemite tuvastus ei tööta araabia ja prantsuse keelega.
- Google Cloud Natural Language [21] – veebipõhine rakendusliides, mis toetab ainult suuremaid keeli, pakkudes neile morfoloogilist töötlust, meelsusanalüüsi ning nimeolemite tuvastust. Kirjeldatud funktsionaalsuse paremaks kohandamiseks erinevatele vajadustele on täpsema keelemudeli saamiseks võimalik andmeid juurde lisada. Teenuse kasutamise hind kujuneb rakendatud funktsioonide tüübi ning tehtud päringute arvu põhjal.

Loomuliku keele töötluse vahendeid leidub hulganisti veelgi, kuid toodud näited demonstreerivad hästi töövahendite mitmekesisust. Olenevalt rakendusalast on seega oluline läbi mõelda täpsed nõuded soovitud tekstitöötluse kohta enne valiku tegemist.

## 1.5 Olemasolevad mobiilirakendused

Kuulutuses oleva info kalendrisse sisestamise automatiseerimise idee ei ole küll uus, kuid universaalset mobiilirakendust selle jaoks eriti lihtne leida ei ole. Olemasolevate rakenduste põhiliseks kitsaskohaks on toetatud keelte väike hulk, aga raskust valmistab ka tuvastatud tekstist õige info leidmine. Järgnevalt on lühidalt kirjeldatud kahte katsetatud mobiilirakendust ning nende eeliseid ja puudujääke.

- SnapEvent [22] – rakendus Android platvormile. Võimaldab telefoni kaameraga pilti teha, millelt püütakse tuvastada ürituse nimi, aeg ja toimumiskoht. Rakendusevälisеid pildifaile kasutada ei ole võimalik. Enne kalendrisse salvestamist on kasutajal võimalik info üle kontrollida ning vajadusel muudatusi teha. Rakendus lõikab ilma selge põhjuseta ära pildi ülemise ja alumise ääre, mistõttu võib oluline osa analüüsist välja jääda. Kui kuupäeva tuvastamine ebaõnnestub, siis kalendrit sündmuse sisestuseks ei avata. Toetab ainult ingliskeelseid kuulutusi. Põgusa katsetamise tulemusel oli tuvastuse täpsus pigem madal (tihti olid näiteks tuvastatud kuupäev ja toimumiskoht valed).

- Google Lens [23] – rakendus Android ja iOS platvormidele. Lisaks muule funktsionaalsusele pakub ka võimalust fotolt tuvastada kalendrisündmusi. Protsess ei ole alati täiesti automaatne, sest nõuab mõnikord pildilt kuupäeva ja/või kellaja käsitsi selekteerimist, enne kui kalendrisse lisamise võimalust pakutakse. Tuvastatakse ürituse pealkirja, toimumisaega ja -kohta. Ametlikult eesti keelt ei toetata. Põgusa katsetamise tulemusel oli tuvastuse täpsus pigem hea, kuid tihti jäi genereeritud kalendri sissekandest välja oluline info (ürituse pealkiri ja toimumiskoht).

Olulise puudujäägina jääb rakendusi analüüsides silma eesti keele toe puudumine. Antud töö raames valmiva mobiilirakenduse eesmärk on seega luua rakendus, mis saab ennekõike hakkama eestikeelsete kuulutuste töötlemisega. Lisaks on kasutusmugavuse poole pealt oluline, et kasutaja saaks kasutada nii varasemalt tehtud pilte kui ka teha uusi pilte. Samuti peab rakendus olema võimalikult automaatne ning tuvastama kuulutuselt olulised osad ilma kasutaja abita. Mobiilirakendus on piisavalt täpne siis, kui säästab manuaalse sisestuse meetodiga võrreldes kasutaja aega.

# 2. Valminud lahenduse kirjeldus

Lõputöö tulemusena valmis mobiilirakendus Androidi operatsioonisüsteemile [24] koos veebipõhise rakendusliidesega [25]. Loodud mobiilirakenduse eesmärk on kuulutusest tehtud pildilt tuvastada tekst, saata see edasiseks analüüsiks veebiteenusele ning sealt saadud vastuse põhjal lisada kuulutuselt leitud andmed kasutaja kalendrisse (nt rakendusse Google Calendar [26]). Veebipõhise rakendusliidese eesmärk on loomuliku keele töötluse vahendeid kasutades tuvastada päringuga saabunud tekstist nimeolemid ning need päringu teinud rakendusele tagastada. Järgnevas kahes alampeatükis antakse valminud mobiilirakendusest ja rakendusliidesest detailsem ülevaade.

## 2.1 Mobiilirakendus

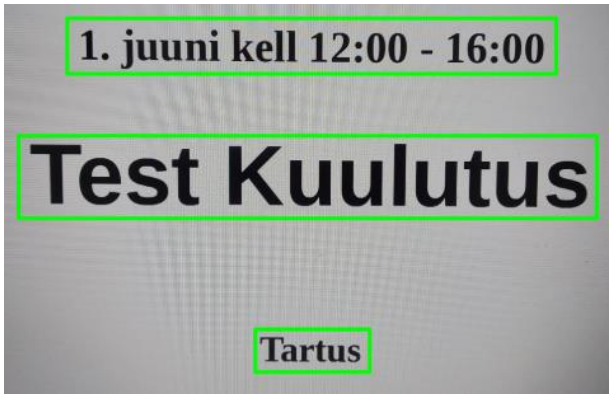
Kuulutuses oleva info automaatseks kalendrisse sisestamiseks peab mobiilirakendus võimaldama automatiseerida kogu protsessi võimalikult suurel määral. Ülesande eelduseks on kasutaja poolt sisendina antud pildifail (ürituse kuulutus). Püstitatud eesmärk on edukalt täidetud, kui kasutajale kuvatakse rakenduse töö lõppedes võimalikult täpne kalendri sissekanne.

### 2.1.1 Arhitektuur

Kasutajalt sisendi saamiseks annab mobiilirakendus võimaluse teha telefonikaameraga uus foto või valida olemasolev foto seadmesse salvestatud pildifailide seast. Uue foto tegemiseks küsib mobiilirakendus esmalt kasutajalt vajalikud load: ligipääs kaamerale ja kasutaja failidele. Olemasolevate pildifailide kasutamiseks rakendus erilube ei vaja.

Järgmiseks tuvastatakse pildilt kogu tekst. Teksti tuvastamine toimub nutitelefonis lokaalselt. Kuna kuulutus võib oma ülesehituselt olla väga ettearvamatu, siis pildile mingit eeltöötlust ei tehta. Kui tekstituvastus mingil põhjusel ebaõnnestub, annab rakendus sellest kasutajale teada.

Tuvastatud tekst jaguneb väiksemateks fragmentideks. Eraldi fragmenti kuulub tekst, mis visuaalselt moodustab ühe terviku kirja suuruse, tüübi ja asukoha alusel (joonis 1). Leitud fragmendid järjestatakse asukohapõhisel printsiibil vasakult paremale, ülevalt alla.




Joonis 1. Pildilt tuvastatud teksti jaotumine väiksemateks fragmentideks (piiritletud rohelise kastiga).

Kuulutuses oleva info automaatseks kalendrisse sisestamiseks tuleb esmalt jaotada tuvastatud tekstifragmendid huvipakkuvatesse kategooriatesse. Nendeks on ürituse

- pealkiri,
- toimumiskoht,
- toimumisaeg (alguskuupäev ja -kellaaeg, valikuliselt ka lõppkuupäev ja -kellaaeg).

Tekstifragmendid, mis ühegi eelnimetatud kategooria alla ei sobitu, tuleb välja filtreerida.

Tekstist olulise teabe eraldamiseks on mitmeid viise. Näiteks võib igale tuvastatud tekstifragmendile püüda ennustada kategooriat tema asukoha, suuruse või mõne muu omaduse alusel. Leidub aga ka mitmeid loomuliku keele töötluse vahendeid, mis võimaldavad tuvastada nimeolemeid. Parima tulemuse saavutamiseks on töös kasutatud nende meetodite kombinatsiooni.

Kuna ürituse tüüpe on väga erinevaid, siis tihtipeale võib olla keeruline piiritleda, milline on täpselt ürituse korrektne pealkiri. Ka inimese enda eelistus mängib rolli, kuidas kalendrisündmuseid üles märgitakse, vajadusel ümber sõnastatakse jne. Seetõttu valitake ürituse pealkirjaks alati kuulutuse pealt leitud suurima kirjaga tekst. Selline lahendus ei garanteeri küll igakord adekvaatset tulemust, kuid on loogiline eeldada, et mõõtmetelt suur tekst sisaldab endas piisavalt olulist informatsiooni inimeste tähelepanu tõmbamiseks ning sobib seega ka üritust kirjeldama. Samuti on selline lähenemisviis lähtekeelest sõltumatu võimaldades pealkirja määrata ükskõik mis keeles olevale kuulutusele.

Toimumiskoha ja -aja kindlaks määramiseks kasutatakse nimeolemite tuvastamise vahendeid. Kuigi harilikult ei moodusta kuulutusel olev tekst täielikult sidusat teksti, siis on sellegipoolest võimalik loomuliku keele töötluse abil mainitud informatsiooni mugavalt

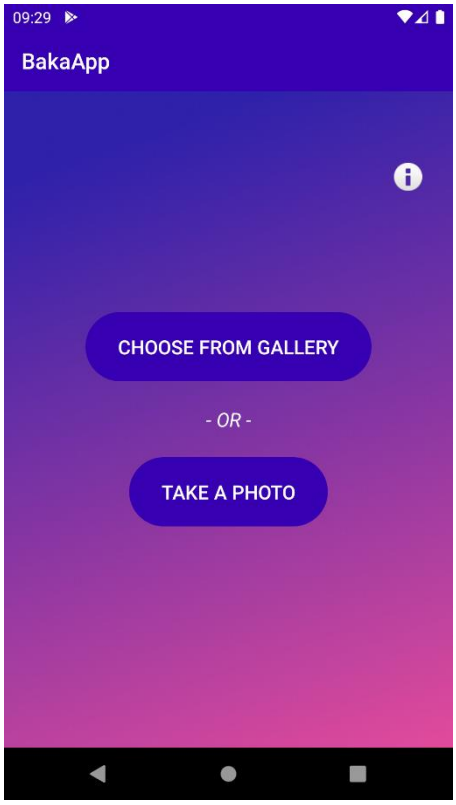
kätte saada. Toetatud keelte hulka on võimalik kiiresti suurendada, kui leidub vastavas keeles nimeolemite tuvastuseks sobivaid keelemudeleid.

Nimeolemite ja ajaväljendite tuvastamiseks pöördub mobiilirakendus veebi teel rakendusliidese poole. Päringu kokkupanemiseks moodustatakse kõigepealt leitud tekstifragmentidest üks tervik liites kokku kõik tuvastatud fragmendid. Tekstifragmendid reastatakse tuvastamise järjekorras (kuulutuselt vasakult paremale, ülevalt alla) ning eraldatakse üksteisest tühikuga. Eduka päringu korral tagastab server analüüsi tulemused, milles sisaldub tuvastatud algus- ja lõppaeg ning toimumiskoht (kui need sisendi põhjal tuvastada õnnestus). Kui pildilt tuvastati mitu asukohta, siis valitakse toimumiskohaks nendest esimene. Algus- ja lõppaeg on alati üheselt määratud ning nende töötlemisega mobiilirakendus tegelema ei pea.

Lõpuks avatakse kasutaja jaoks vaikimisi seadistatud kalendrisündmusi haldav rakendus. Sellega koos kuvatakse uue sündmuse lisamise vaade, mis on täidetud eelnevate sammude tulemusel kogutud informatsiooniga. Väljadele, millele väärtust leida ei õnnestunud, ning kõikidele ülejäänud parameetritele omistab kalendrirakendus vaikeväärtuse. Sel viisil saab kasutaja andmed kalendri sissekandes enne salvestamist üle kontrollida ning vajadusel käsitsi muudatusi teha. Samuti saab kasutaja ürituse kalendrisse lisamisest loobuda.

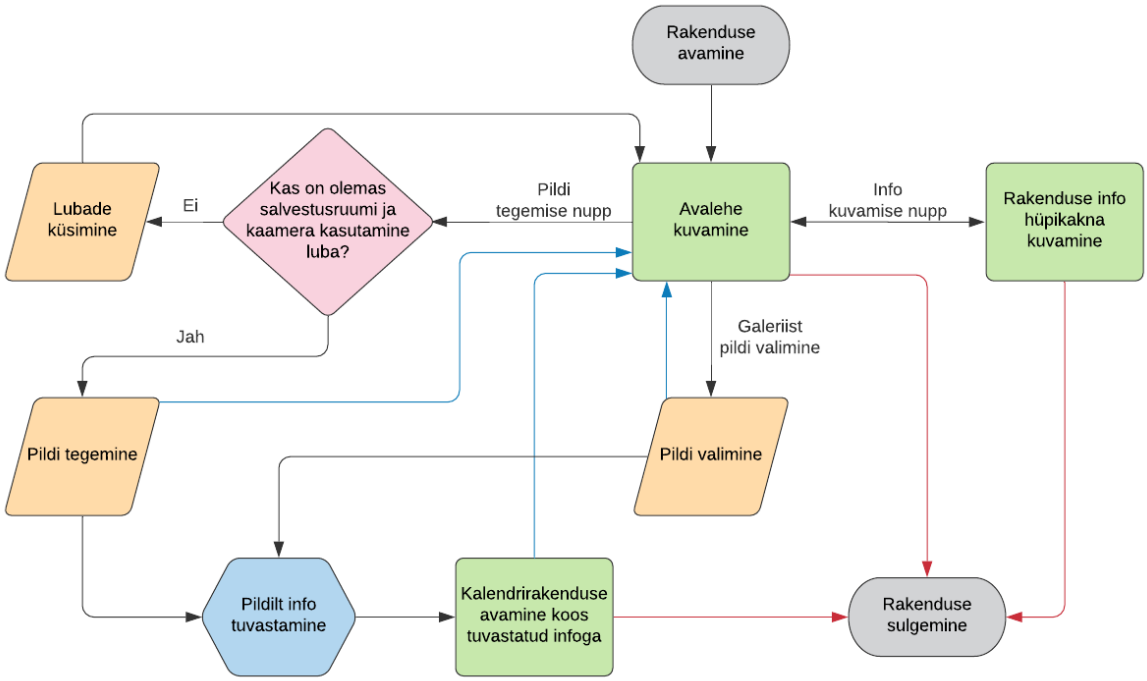
### 2.1.2 Rakenduse kasutamine

Mobiilirakendusele on loodud minimalistlik graafiline kasutajaliides, mille eesmärk on lihtsustada kasutaja suhtlust rakenduse erinevate funktsioonidega. Rakendusel on vaid üks vaade (joonis 2), millel kuvatakse kolm nuppu. Paremas ülaservas olev info-nupp avab hüpikteate mobiilirakenduse tutvustuse ning kasutusjuhistega. Kasutajalt sobival kujul sisendi saamiseks on olemas nupud kaameraga kuulutusest pildi tegemiseks ning telefoni galeriist pildi valimiseks. Pildi töötlus toimub taustal ning sel ajal kuvatakse kasutajale laadimise indikaatorit, samaaegselt uut pilti teha või valida ei ole võimalik. Analüüsi lõppedes avatakse automaatselt kalendrirakendus leitud infoga. Kalendrirakenduse tüüp sõltub süsteemsetest parameetritest ning kasutaja seadetest.




Joonis 2. Loodud mobiilirakenduse avaleht.

Mobiilirakenduse kasutusskeem on visualiseeritud joonisel 3. Halli taustaga on märgitud rakenduse avamise ja sulgemise olekud. Roheline värv tähistab kasutajaliideses kuvatava vaate muutusi ja kollane toon kasutajalt sisendi küsimist. Vooskeemi roosa taustaga element märgib rakenduse kasutamiseks vajalike lubade kontrolli ning sinist värvi element taustal toimuvat kuulutuse pildi töötlusprotsessi. Nooled markeerivad olekute vahelisi üleminekuid. Sinise värviga esile toodud nooled märgivad nutitelefoni „Tagasi“ nupu vajutamist ning punased nooled „Kodu“ nupu vajutamist.




Joonis 3. Mobiilirakenduse kasutamise vooskeem.

Mobiilirakenduse funktsionaalsuse täismahus kasutamiseks on vajalik internetiühendus. Ühenduse puudumise korral kasutajat teavitatakse sellest ning rakendus on võimeline tööd jätkama. Sel juhul on võimalik kuulutuselt tuvastada vaid ürituse pealkirja.

### 2.1.3 Kasutatud tehnoloogiad

Mobiilirakendus on loodud kasutamiseks Androidi operatsioonisüsteemiga nutitelefonis. Minimaalne toetatud Androidi versioon on 4.4 (KitKat). Arendamisel on seega silmas peetud ühilduvust Androidi tarkvaraarendus komplekti (SDK) versiooniga 19. Mobiilirakenduse praegune sihtversioon on aga Android 10, millele vastab Androidi tarkvaraarendus komplekti versioon 29. Seega on 2020. aasta aprilli seisuga toetatud ligikaudu 98,1% kõikidest Androidi seadmetest [27].

Optilise märgituvastuse teostamiseks kasutatakse Google Mobile Vision API alla kuuluvat Text Recognition rakendusliidest. Kuna tegu on vabavaralise vahendiga, mis suudab hästi tuvastada teksti ka suure müraga piltidelt, sobib see piisavalt hästi kuulutuse pildilt teksti tuvastamiseks. Rakendusliides pakub funktsionaalsust ka tuvastatud tekstifragmentide dimensioonide leidmiseks, mida kasutatakse ürituse pealkirja leidmisel (kuulutuselt tuvastatud suurima kõrgusega tekst).

Veebipõhise rakendusliidesega suhtlemiseks mõeldud klient ja andmeedastusobjektid genereeritakse vabavaralise koodigeneraatoriga Swagger Codegen [28] programmeerimiskeelele Java. Koodi genereerimisel on aluseks veebipõhise rakendusliidese poolt defineeritud teenuse formaalne kirjeldus (lisa 1).

## 2.2 Veebipõhine rakendusliides

Vajalikud tekstitöötlusfunktsioonid on eraldatud omaette rakendusliidesesse, millega mobiilirakendus võrgu teel suhelda saab. Delegeerides tekstitöötlusega seotud töö veebiteenusele, kasutab mobiilirakendus vähem nutitelefoni enda ressursse. Samuti ei pea mobiilirakenduse arendamisel arvestama konfliktidega, mis võivad tekkida, kui loomuliku keele töötluse teegid juhtuvad kasutama erisuguseid programmeerimiskeeli või sõltuvuste versioone.

### 2.2.1 Arhitektuur

Loodud veebipõhine rakendusliides on defineeritud kasutades OpenAPI 3 standardit. OpenAPI 3 standard on vahend REST-põhise teenuse kirjeldamiseks YAML- või JSON-formaadis [29]. Standardi järgimine võimaldab rakendusliidese funktsionaalsust hästi dokumenteerida ning on kergesti laiendatav. Rakendusliidese programmeerimisel on valitud paradigmal REST (*Representational State Transfer*) põhinev arhitektuuristiil, kuna seda iseloomustab ühtlane liides, kliendi ja serveri koodi eraldatus, olekuta päringud ning hea jõudlus ja mastabeeritavus [30].

Rakendusliidesel on kasutamiseks defineeritud kolm otspunkti:

- /estnlp/ner
    - Võimaldab nimeolemite ja ajaväljendite tuvastust eesti keeles.
- /genericnlp/ner
    - Võimaldab nimeolemite ja ajaväljendite tuvastust inglise keeles. Kui sisend ei ole inglise keeles, siis proovitakse seda kõigepealt tõlkida inglise keelde.
- /autonlp/ner
    - Sisendteksti keel tuvastatakse automaatselt ning nimeolemite ja ajaväljendite tuvastamine toimub vastavalt keelele sobivate vahenditega. Seda otspunkti kasutab ka loodud mobiilirakendus.

Kõigile kolmele otspunktile saab saata POST päringu. Päringu keha peab olema JSON-tüüpi jada kujul, mille elementideks on tekstifragmendid. Tekstifragmente analüüsitakse eraldi nende esinemise järjekorras, kuid tagastatud vastuses on tulemused kokku liidetud üheks JSON-tüüpi objektiks. Vastus koosneb võti-väärtus paaridest, kus on kirjas kõik tuvastatud nimeolemid ning ürituse algus- ja lõppaeg (kui need leiti). Täpne struktuur ja tuvastatavate nimeolemite hulk varieerub olenevalt otspunktist (lisa 1).

Näidispäring:

```
curl
   --header "Content-Type: application/json"
   --request POST
   --data '["sopran Mari Maasikas",
            "suur kontsert Hiiu Pubis",
            "13.04 kell 20.00-22.00 Tallinnas"]'
   https://example.com/autonlp/ner
```

Näidispäringu vastus:

```
{
   "per": ["Mari Maasikas"],
   "org": ["Hiiu pubi"],
   "loc": ["Tallinn"],
   "start": "2020-04-13T20:00:00",
   "end": "2020-04-13T22:00:00"
}
```

Loodud veebiteenust iseloomustab süsteemi kerge hallatavus ja lihtne laiendatavus, aga ka sujuv liidestus teiste süsteemidega ning mitmete dokumenteerimist puudutavate toimingute automatiseerimine. Erinevalt näidispäringust, mis demonstreerib rakendusliidese laiemat kasutusala, saadab loodud mobiilirakendus alati üheelemendilise tekstifragmentide jada (st kogu kuulutuselt tuvastatud tekst on ühes fragmendis). Samuti läheb mobiilirakendusel lõpuks vaja vaid tuvastatud algus- ja lõppkuupäeva ning asukohaga seotud infot.

### 2.2.2 Kasutatud tehnoloogiad

Konkreetsele probleemile kõige sobivamat töövahendit valides tuleb arvestada mitme aspektiga. Esiteks peab loodav rakendus kindlasti toetama eesti keelt, et see oleks eestikeelsete kuulutustega kasutatav. Võõrkeeltest peab rakendus minimaalselt toetama inglise keelt. Teiseks on oluline, et rakenduse kasutamine ja testimine ei oleks piiratud päringute mahu või sellega kaasneva hinna poolest. Seetõttu tuleb võimalusel eelistada

vabavaralisi variante. Kuna lõputöö kirjutamise hetkel ei ole autorile teadaolevalt veel ühtegi kõikehõlmavat varianti, mis kataks piisava funktsionaalsusega ära ka väiksemad maailma keeled nagu seda on eesti keel, tuleb lahenduse välja töötamisel kombineerida erinevaid saadaval olevaid teeke. Järgnevalt kirjeldatakse tehtud tehnoloogilisi valikuid.

Eestikeelsetelt kuulutustelt info leidmiseks rakendatakse Pythoni teeki EstNLTK (versioon 1.4.1). Ürituse asukoha määramiseks kasutatakse nimeolemite tuvastamise funktsionaalsust (objekti *Text* atribuuti *named_entities*), mis muuhulgas oskab tekstist leida asutusi ning geograafilisi asukohti [31]. Kuigi etteantud tekstist võidakse leida mitmeid nimeolemeid, siis loodud mobiilirakendus kasutab neist lõpuks ainult esimesena leitut (eelisjärjekorras asutust ja seejärel geograafilist asukohta). Rakendusliidese universaalsuse mõttes tagastatakse siiski kõik leitud nimeolemid.

Ürituse toimumisaja leidmiseks kasutatakse EstNLTK teegi ajaväljendite tuvastamise funktsionaalsust (objekti *Text* atribuuti *timexes*), mis tagastab kõik leitud ajaväljendid TIMEX-vormingus [32]. Toimumisaja määramisel arvestatakse, et algusaeg peab olema enne lõppaega. Kui leitakse ainult üks ajaväljend, siis arvestatakse see nii algus- kui ka lõppajaks (kui leiti ka kellaaeg, siis määratakse ürituse kestvuseks vaikimisi tund aega). Rohkem kui kahte esimesena leitud kuupäeva ja kellaaega ei käsitleta kunagi. Kui aastaarvu ei ole kuulutusel välja toodud või tegemist on minevikus toimunud üritusega, siis märgib rakendus toimumisajaks jooksva aasta numbri. Selleks, et kuupäevade tuvastus töötaks paremini, asendatakse regulaaravaldise abil kõik numbrilisel kujul olevad kuud vastavate kuunimetustega. Numbripaare, mille vahel esineb koolon, peetakse kellaaegadeks, ning neile lisatakse juurde märksõna „kell“. Kuigi taoline töötlus ei ole alati vajalik, aitab see vähendada riski, et ajaväljendite leidmise funktsioon satub ebapiisava konteksti tõttu segadusse ja ei leia tekstifragmendist kuupäeva või kellaaega üles.

Ingliskeelsetelt kuulutustelt info leidmiseks rakendatakse Pythoni teeki spaCy (versioon 2.0). Nii ürituse asukoha kui ka toimumisaja määramiseks kasutatakse teegi nimeolemite tuvastamise funktsionaalsust (objekti *Doc* atribuuti *ents*), mis tagastab muuhulgas leitud ajaväljendid ning ka eriliiki asutused ja geograafilised asukohad [33]. Algus- ja lõppaja täpsemaks määramiseks kasutatakse eelmises lõigus kirjeldatud metoodikat.

Keele tuvastuseks on kasutusel Pythoni teek TextBlob (versioon 0.15.3) [34]. Sisendteksti keele määramine on oluline, et suunata eestikeelse teksti töötlemise ülesanne EstNLTK teegile, muud juhud aga spaCy teegile. Juhul kui sisendtekst ei ole ei eesti ega inglise keeles,

püütakse TextBlob teegi abil tõlkida sisend inglise keelde[1] ning jätkata saadud ingliskeelse mudeli põhjal nimeolemite tuvastust. Ühest keelest teise tõlkime vähendab küll tulemuste täpsust, sest sel juhul ei pruugi keelemudel osata tuvastada näiteks väiksemaid kohanimesid, kuid võimaldab siiski rakendusel tööd jätkata.

Mainitud teekide kasutamiseks programmeeritakse rakendusliidese implementatsioon keeles Python (versioon 3.5). Serveripoolne koodi tüüpsisu genereeritakse Flask veebiraamistikule vabavaralise koodigeneraatoriga Swagger Codegen [28]. Kuna rakendusel on väga palju erinevaid sõltuvusi, on ta mugavamaks ja sõltumatumaks juurutamiseks paigutatud konteinerisse kasutades töövahendit Docker [35].

[1] TextBlob kasutab tõlkimisesks Google Translate rakendusliidest, mis toetab üle 100 keelepaari. Täpsemalt vt nt: https://cloud.google.com/translate (08.05.20)

# 3. Tulemused

Töö tulemusena valmis mobiilirakendus, mis võimaldab kasutajal automaatselt teha kalendrisse sissekanne kuulutusest tehtud pildi põhjal. Selles peatükis kirjeldatakse rakenduse testimise metoodikat ning analüüsitakse saadud tulemusi. Samuti antakse hinnang rakenduse täpsusele ja kasutusmugavusele ning pakutakse välja edasiarendamise võimalusi.

## 3.1 Testimine

Testimise eesmärk on anda lähtepunkt loodud mobiilirakenduse täpsuse hindamiseks. Kuna ürituse tüüpe ja kuulutuse disaine on väga mitmesuguseid, ei pruugi igale kuulutuse pildile ainuõiget väljundit alati leiduda. Näiteks võib kontsertturneed reklaamivale üritusele sobida pealkirjaks nii kontsertturnee enda nimetus kui ka esineja nimi. Seetõttu oleneb väljundi täpsus ja sobivus tihti kasutaja enda subjektiivsest arvamusest.

Paremini võrreldavate tulemuste saamiseks otsustati seega mõõta sündmuse kalendrisse lisamiseks kuluvat aega kahel viisil. Kõigepealt mõõdetakse, kui kaua läheb aega selle tegevuse sooritamiseks loodud mobiilirakendust kasutades. Seejärel mõõdetakse, kui palju võtab aega sama sündmuse käsitsi kalendrisse lisamine. On loogiline eeldada, et mida adekvaatsem on rakenduse väljund, seda rohkem säästab kasutaja aega käsitsi sisestamise arvelt. Samas piisavalt vigase väljundi korral võib vigade parandusele ja puudujääkide kõrvaldamisele kuluda kokkuvõttes rohkem aega kui sündmuse käsitsi lisamisele.

Mobiilirakendust testisid kirjeldatud viisil kokku neli inimest. Sobiliku teststsenaariumi loomisel tuli arvestada, et erinevate kasutajate testimise tulemuste paremaks võrdlemiseks peab rakendus saama töötlemiseks sarnastes tingimustes tehtud kuulutuse pildi. Ühtlase kvaliteedi tagamiseks on seega testitavad kuulutused eelnevalt valmis pildistatud.

Testitavate kuulutuste hulga moodustas viis telefonikaameraga tehtud pilti (lisa 2). Kõik kuulutused pildistati sama seadmega sülearvuti ekraanilt. Viiest kuulutusest neli olid eesti keeles ning üks inglise keeles. Valim koostati selliselt, et tulenevalt kujunduslike elementide iseärasusest saaks testida nii lihtsama ja selgema ülesehitusega pilte kui ka keerulisemaid ja mürarohkemaid postreid. Esimene testitav kuulutus oli käsitsi loodud ning sellel kujutati üksnes olulist infot musta tekstina valgel taustal. Ülejäänud neli pärinesid erinevatest internetiallikatest võimaldades seega hästi testida rakenduse täpsust ka reaalsete kuulutuste peal.

Iga valimisse kuuluva kuulutuse testimise protsess toimus kahes etapis. Esimeses etapis testiti loodud mobiilirakenduse abil kalendrisse sündmuse lisamise kiirust järgnevalt kirjeldatud sammude alusel.

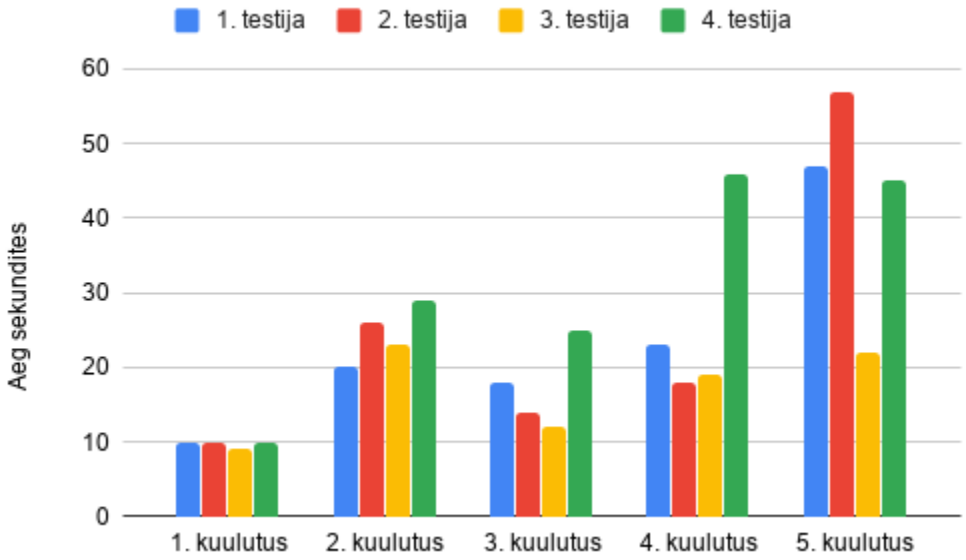
1. Kasutajal on telefonisse salvestatud testitava kuulutuse pilt ning rakenduse avaleht on avatud.
2. Kasutaja käivitab stopperi.
3. Kasutaja vajutab pildigaleriist valimise nuppu.
4. Kasutaja otsib kuvatud pildifailide seast üles testitava kuulutuse ning vajutab sellel.
5. Kasutaja ootab, kuni avatakse kalendrirakendus koos uue sündmuse lisamise vaatega.
6. Kasutaja kontrollib, kas lisatava sündmuse pealkiri, toimumisaeg ja toimumiskoht on sobivalt väärtustatud, ning vajadusel teeb parandusi.
7. Kasutaja vajutab kalendrirakenduses sündmuse salvestamise nuppu.
8. Kasutaja peatab stopperi ja märgib üles kulunud aja.

Teises etapis testiti käsitsi sündmuse kalendrisse lisamise kiirust. Aluseks võetud sammud on kirjeldatud järgmiselt.

1. Kasutajal on avatud kalendrirakendus.
2. Kasutaja käivitab stopperi.
3. Kasutaja navigeerib uue kalendrisündmuse lisamise nupuni ja vajutab seda.
4. Kasutaja sisestab sobilikud väärtused ürituse pealkirja, toimumisaja ja toimumiskoha väljadele.
5. Kasutaja vajutab kalendrirakenduses sündmuse salvestamise nuppu.
6. Kasutaja peatab stopperi ja märgib üles kulunud aja.

Testimise esimese etapi tulemused (lisa 3) on esitatud joonisel 4. Diagrammilt on näha, et üldine trend on kuulutuste lõikes sarnane. Esimese kuulutuse töötlemine loodud mobiilirakendusega võttis kõigil katses osalenutel kõigest 9-10 sekundit. Kuna tegu on kõige selgema struktuuriga postriga, millest info tuvastamine on rakendusel lihtne, siis on ka saadud tulemus ootuspärane. Ka teise kuulutuse töötlemiseks kulunud aeg oli testijate lõikes sarnane. Teisel kuulutusel oli esimesega võrreldes palju pikem pealkiri ning samuti märkis rakendus pealkirja ekslikult liiga palju infot. See võib olla põhjuseks, miks kasutajapoolne andmete valideerimine võttis seekord rohkem aega. Olulise info rohkus iseloomustab ka kolmandat kuulutust, mistõttu pidid kasutajad üle kontrollima rohkem

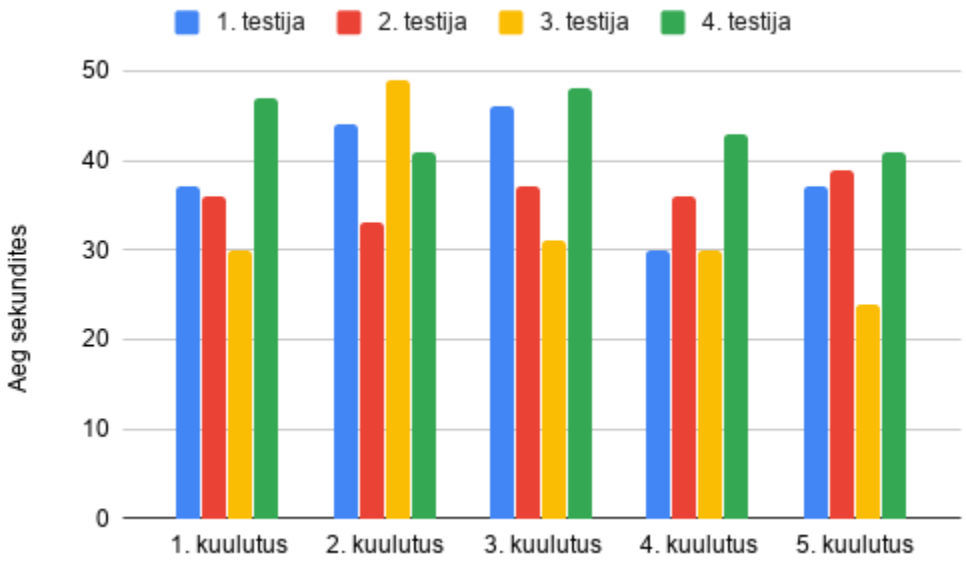
andmeid võrreldes esimese kuulutusega. Neljandal postril esines kõige rohkem infomüra ning kasutatud oli raskesti loetavat kirja. Samuti oli keeruline piiritleda, milline on sobiv pealkiri üritusele. Inimeste eelistuste erinevus tuleb ka jooniselt hästi esile, sest neljandal testijal kulus teistega võrreldes umbes kaks korda rohkem aega kalendrisündmuse õigele kujule saamiseks. Esimene, teine ja kolmas testija aga eelistasid lühemat ja ebatäpsemat ürituse kirjeldust, mistõttu kulus neil ka kokkuvõttes vähem aega. Viimase viienda postri töötlemine oli rakenduse jaoks kõige keerulisem, mistõttu kasutajad pidid kindlasti parandusi tegema nii ürituse nimes, toimumisajas kui ka -kohas. Sellest tulenev lisakulu on ka põhjus, miks kasutajatel läks viimase ürituse andmete kalendrisse saamise peale keskmisest rohkem aega.




Joonis 4. Mobiilirakenduse abil kalendrisse sündmuse lisamiseks kulunud aeg.

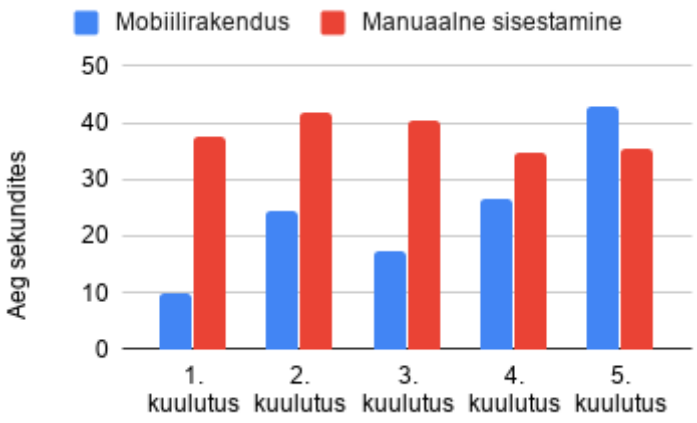
Testimise teise etapi tulemused (lisa 3) on toodud välja joonisel 5. Diagrammilt on näha, et erinevate kuulutuste käsitsi kalendrisse sisestamisele kulunud aeg on palju ühtlasema jaotusega võrreldes eelmises etapis leituga. Kõikidel katses osalenutel inimestel aitas nutitelefon trükivigu automaatselt parandada ja pakkus sõnasoovitusi, mis muutis teksti sisestamise oluliselt lihtsamaks ning kiirendas kogu protsessi. Samas oli toimumisaja määramine küllaltki aeganõudev tegevus, sest näiteks esimese kuulutuse korral oli tarvis märkida eraldi nii algus- kui ka lõppaeg ning kummalgi juhul ka õige kuu, päev ja kellaaeg.

Mainitud tegurite kombinatsiooni arvestades on ootuspärane, et väga suurt lahknevust erinevatele kuulutustele kulunud aja suhtes ei tekkinud.

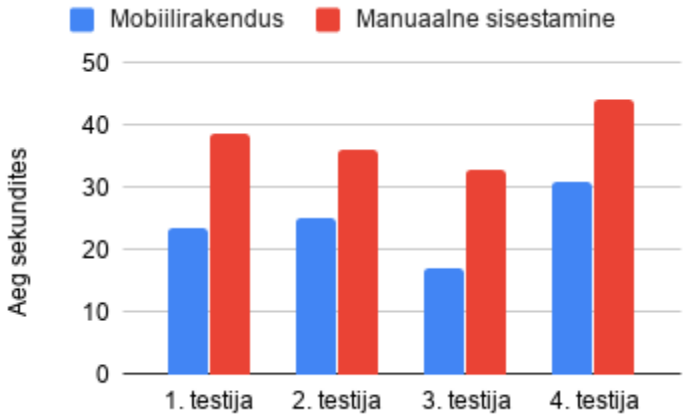



Joonis 5. Käsitsi sündmuse kalendrisse lisamiseks kulunud aeg.

Joonisel 6 on esitatud iga kuulutuse töötlusele kulunud aeg aritmeetilise keskmisena, mis on arvutatud kõigi nelja testija tulemuste peale kokku. Diagrammilt on näha, et esimese nelja kuulutuse kalendrisse sisestamine oli loodud mobiilirakenduse abil kiirem kui sama info käsitsi sisestamine. Ainult viimase viienda postri puhul oli mobiilirakendus käsitsi meetodist aeglasem (vastavalt 42,75 sekundit ja 35,25 sekundit). Hästi joonistub välja ka käsitsi sisestamisel saadud ühtlased tulemused. Näiteks on esimesele kuulutusele kulunud aeg (37,5 sekundit) väga sarnane viiendale kuulutusele kulunud ajaga (35,25 sekundit). Suurim erinevus on teisele ja neljandale kuulutusele kulunud aeg, mille vahe on 7 sekundit. See on aga kontrastiks esimese etapi tulemustega, kus viienda kuulutuse töötlemisele kulus keskmiselt 33 sekundit rohkem kui esimese kuulutuse töötlemisele. Seega tuleb jooniselt paremini esile ka mobiilirakendusega saavutatud tulemuste kõikumine. Ebastabiilsus tulenes testitavate kuulutuste mitmekesisusest, mistõttu pidid kasutajad olenevalt rakenduse väljundist rohkem või vähem parandusi tegema.




Joonis 6. Kuulutuste töötlemisele keskmiselt kulunud aeg.




Joonis 7. Keskmiselt ühe kuulutuse töötlemiseks kulunud aeg testijate lõikes.

Tulemuste veel paremaks ilmestamiseks on joonisel 7 esitatud ka iga testija keskmiselt kulutatud aeg kõikide kuulutuste peale kokku. Aritmeetilised keskmised on arvutatud nii testimise esimese etapi kui ka teise etapi lõikes. Kokkuvõttes selgub, et võrreldes käsitsi sisestamisega kulus kõigil katses osalenutel mobiilirakenduse abi kasutades keskmiselt vähem aega. Esimene testija säästis 15,2 sekundit, teine 11,2 sekundit, kolmas 15,8 sekundit ja neljas 13 sekundit. Keskmistades joonisel 7 toodud tulemused veel omakorda, kulus testijatel esimeses etapis keskmiselt kokku 24,15 sekundit ja teises etapis 37,95 sekundit. See tähendab, et keskmine ajavõit oli kogu testimisel saadud tulemuste põhjal 13,8 sekundit.

## 3.2 Täpsuse hindamine

Eelmises peatükis kirjeldatud testvalimis leidus erineva karakteristikaga kuulutusi, millest tulenevalt ei olnud töötluse tulemused alati ühteviisi edukad. Loodud mobiilirakenduse täpsuse hindamiseks on seega sobiv analüüsida testimisel kasutatud kuulutuste näiteid (lisa 2).

Esimese kuulutuse tuvastamisega sai rakendus hästi hakkama. Kuulutuse pealkirjaks määrati „Test Kuulutus“ ning asukohaks „Tartu“. Ka sai rakendus edukalt hakkama sündmuse kestvuse tuvastamisega, mis kujutas endast konkreetselt defineeritud ajaperioodi. Algusajaks määrati korrektselt „1. juuni 2020 kell 12:00“ ja lõppajaks „1. juuni 2020 kell 16:00“. Kuna tegu oli selges šriftis musta tekstiga valgel taustal, oli nii automaatne märgituvastus kui ka sellele järgnev info eraldamine võrdlemisi lihtne.

Teine kuulutus oli testvalimi ainus ingliskeelne näide. Kuulutusel oli sinistes toonides taust ja valge tekst. Kuna kuulutusel olev tekst oli üsna ühtlase kõrgusega, siis märkis rakendus ürituse pealkirjaks kogu leitud teksti: „Webinar on Academic Writing „Tools for Writers“ 22.04.2020 at 14:00“. Kuigi kuupäev ja kellaeg on üleliigne, on siiski märkimisväärne, et rakendus tuvastas kogu teksti ilma ühegi veata. Sarnaselt eelmisele kuulutusele oli siingi toimumisaeg väga selge struktuuriga, mistõttu märkis rakendus ilma probleemideta algusajaks „22. aprill 2020 14:00“ ja lõppajaks „22. aprill 2020 kell 15:00“. Sündmuse kestvuseks määrati vaikimisi üks tund, sest antud oli vaid algusaeg. Sündmuse asukohaks märgiti „Webinar“, mida võib kuulutuse kontekstis pidada adekvaatseks.

Kolmanda kuulutuse töötlemise tulemusel pani rakendus üritusele pealkirjaks „KONTSERTAKTUS“. Kuna kuulutusel esines see sõna suurte tähtedega, siis sisestas ka rakendus selle kalendrisse samal kujul. Ootuspäraselt jäi pealkirjast välja üritust täpsemini kirjeldav osa „Eesti Vabariigi 102. aastapäeva“, sest võrreldes kõrgeima leitud tekstifragmendiga oli selles fragmendis esinevate tähtede kõrgus palju väiksem. Toimumisaeg tuvastati korrektselt ning algusajaks määrati „21. veebruar 2020 kell 14:00“ ja lõppajaks „21. veebruar 2020 kell 15:00“. Asukohaks märgiti „Tartu Ülikool aula Eesti Vabariik“, mis põhimõtteliselt annab küll õige info edasi, aga demonstreerib samas hästi, et näiteks ürituse pealkirjas leiduvaid kohanimesid võib rakendus ekslikult pidada ürituse enda toimumiskohaks.

Neljandal kuulutusel oli kõige keerulisemaks osaks ürituse pealkirja tuvastamine, mis oli trükitud kunstipärases šriftis ning sisaldas muuhulgas bändi logo („AC/DC Highway

Symphony“). Rakendus selle tuvastamisega hästi hakkama ei saanud ja märkis pealkirjaks „2Higha yipho“, mida ei saa kuidagi lugeda sobivaks. Samas kontserdi toimumiskoht ja -aeg tulid kuulutuselt selgemini esile, mistõttu suutis rakendus õigesti märkida algusajaks „22. september 2020“ ja lõppajaks „22. september 2020“ (kuna lõppaega ega kellaaegu polnud eraldi märgitud, siis määrati kestvuseks terve päev) ning toimumiskohaks „Alexela kontserdimaja“.

Viies kuulutus oli kujunduse poolest kõige kunstilisem, kus tekst sulas taustaga nii palju ühte, et rakendus jäi jällegi hätta optilise märgituvastusega. Pealkirjaks märgiti „RESSAAREEPAEVAD“, mida ei saa lugeda korrektseks. Ka toimumisperioodist suudeti tuvastada vaid lõppkuupäev „8. august 2020“, mistõttu märkis rakendus kalendrisse ekslikult, et tegu on ühepäevase üritusega. Toimumiskoha jättis rakendus tühjaks, sest tuvastatud tekst oli liiga poolik ja segane, mille tõttu ei olnud sealt võimalik ka asukohti tuvastada.

## 3.3 Hinnang

Testijate hinnangul on loodud mobiilirakendust lihtne ja mugav kasutada. Positiivsena tuuakse esile rakenduse minimalistlikkust, millega on tagatud kõik vajalik funktsionaalsus ilma ühegi üleliigse elemendita. Samuti on kasutajaliides intuitiivne ning rakenduse pakutavad funktsioonid arusaadavad.

Kuigi tagasisides toodi välja, et keerulisemate kuulutuste tuvastamiseks ei ole rakendus veel piisavalt töökindel, sest halvimal juhul tekitatakse kasutajale käsitsi tööd hoopis juurde, leidsid testijad, et rakendus on siiski piisavalt täpne lihtsamate kuulutuste kalendrisse salvestamiseks. Ka testimise tulemused kinnitavad, et loodud mobiilirakenduse abiga on lihtsamatel juhtudel aja säästmine võimalik. Oluline on siiski märkida, et olenevalt olukorrast võib hea pildi saamiseks kuluda kokkuvõttes rohkem aega, kui seda nõuab ürituse andmete käsitsi sisestamine. Seetõttu ei ole rakendus alati asenduseks muudele viisidele.

Tuginedes testimise tulemustele ja testijate tagasisidele võib seega järeldada, et mobiilirakenduse pakutud ürituse pealkirja, toimumisaja ja -koha andmed on lihtsama ülesehitusega kuulutuse korral piisavalt adekvaatsed. Suurema tausta- ja infomüraga postrite puhul aga teeb rakendus vigu. Kuulutuselt õigete andmete tuvastamise võib keeruline olla mitmel põhjusel. Näiteks mõjutab täpsust sisendina antava pildi kvaliteet. Parema tulemuse saamiseks on oluline pildistada hästi valgustatud ruumis ning jälgida, et kaamera oleks

fookuses ja kuulutusega samal tasandil. Kuigi rakendus saab hakkama väga erinevatelt pindadelt pildistatud kuulutustega, annab paberkujul olevast kuulutusest tehtud foto selgema tulemuse kui sama kuulutuse arvutiekraanilt pildistamine.

Automaatse info tuvastuse muudab samuti keeruliseks kuulutustel leiduv üleliigne või segane info. Näiteks ebaproportsionaalselt suuri sponsorite logosid võib rakendus eksikombel pidada pealkirja või toimumiskoha osadeks. Ka kohanimede korrektne tuvastamine ei ole alati lihtne. Näiteks on loogiline eeldada, et üritus Tartu linna päev toimub Tartus, ent Tartu Noortekoor ei pruugi alati Tartus esineda. Sellises olukorras otsuse langetamise muudab raskemaks ka asjaolu, et mõnikord ei ole kuulutusel eraldi toimumiskohta mainitudki.

Kõige paremini saab loodud mobiilirakendus hakkama ürituse toimumisaja tuvastamisega. Kuna kuupäevad ja kellaajad on üldiselt selge struktuuriga, siis on ka nende leidmine kuulutuselt võrdlemisi lihtne ülesanne. Raskusi tekib ainult väga keerulise kujundusega kuulutuste puhul, kus oluline informatsioon sulandub taustaga üheks, mistõttu ei anna algne optiline märgituvastus piisavalt täpset tulemust edasiseks töötluseks. Ka testijad märkisid oma tagasides, et toimumisaja käsitsi sisestamine on erinevates kalendrirakendustes võrdlemisi tüütu ning loodud mobiilirakendus aitas edukalt seda protsessi muuta kiiremaks ja mugavamaks.

Võrreldes olemasolevate rakendustega toetab selle töö raames loodud mobiilirakendus ainsana eesti keelt. Samuti on ta kõige rohkem ülesandele orienteeritud automatiseerides kogu tuvastamise ja info eraldamise protsessi. Ka testvalimis olevalt lihtsas vormingus ingliskeelselt kuulutuselt tuvastas loodud rakendus andmeid kõige täpsemini. Mobiilirakendus SnapEvent ei saanud ühegi välja tuvastamisega hakkama ning ei avanud seetõttu ka kalendrirakendust. Google Lens oskas kalendrisse märkida ainult ürituse toimumise kuupäeva ja kellaaja. Samas töö raames loodud rakendus suutis leida üritusele piisavalt adekvaatse pealkirja ning ka täpse toimumisaja ja -koha.

Kokkuvõttes võib nentida, et kuigi testimise käigus saavutati väikene ajavõit võrreldes käsitsi sisestamisega, siis praktikas leidub palju selliseid kuulutusi, millega loodud rakendus piisavalt hästi hakkama ei saa. Kui arvestada juurde ka asjaolu, et tavakasutajal tõenäoliselt ei ole vajadust massiliselt üritusi kalendrisse lisada, siis potentsiaalne ajavõit on küllaltki tühine. Rakendusest võivad seega enim kasu saada kasutajad, kelle jaoks käsitsi kalendrisse info sisestamine on tavapärasega võrreldes raskendatud. Samas tõdesid mobiilirakendust

testinud inimesed, et kui rakendus töötaks piisavalt hästi igat tüüpi kuulutusega, siis eelistaksid nad rakenduse kasutamist käsitsi sisestamisele.

## 3.4 Edasiarendamise võimalused

Kuulutuse automaatse töötluse aluseks on võimalikult täpne optiline märgituvastus. Tegu ei ole triviaalse ülesandega, kuna tihti kasutatakse kuulutuste kujundamisel eripärast vormindust ja kirjut tausta. Töö raames kasutatud optilise märgituvastuse vahend sai edukalt hakkama lihtsamat tüüpi kuulutustega, aga jäi hätta näiteks väga vildaka teksti tuvastamisega. Seega üks edasiarendamise võimalus on katsetada võimsamaid optilise märgituvastuse vahendeid täpsuse parandamise eesmärgil. Samuti võib olemasolevaid vahendeid püüda kohandada sobivamaks erinevate masinõppemeetodite abiga.

Tuvastatud tekstist olulise info väljasõelumiseks kasutab rakendus nimeolemite tuvastust. Meetodi täpsust segab tihti ebastandardsel kujul olev sisend, mis ei vasta ortograafia reeglitele. Näiteks on keeruline tuvastada asukohanimesid tekstist, mis on kirjutatud läbinisti suurte tähtedega. Et sellises olukorras paremini hakkama saada, tuleks sobilik nimeolemite tuvastamise mudel ise välja treenida. Mudelite treenimist võimaldavad nii töös kasutatud EstNLTK teek [31] kui ka spaCy teek [36]. Sel viisil saab juurde lisada ka uusi asukoha- ja organisatsiooninimesid, mida praegused mudelid veel tuvastada ei oska.

Rakendus on loodud kasutamiseks eesti- ja ingliskeelsete kuulutustega. Algeline tugi on olemas ka muudele ladina tähestikul põhinevatele keeltele (piirang tuleneb kasutusel olevast optilise tuvastuse töövahendist, mis oskab tuvastada ainult ladina kirja). Eesti kontekstis kohtab aga ka palju venekeelseid kuulutusi (2019. aasta seisuga moodustasid vene rahvusest inimesed Eesti rahvaarvust ligikaudu 25% [37]). Seetõttu oleks hea lisada mobiilirakendusele ka vene keele tugi.

Töö käigus valminud mobiilirakendus ja eraldiseisev veebipõhine rakendusliides on mõlemad kavandatud selliselt, et uue funktsionaalsuse lisamine oleks võimalikult lihtne. Loodud süsteemi eraldatus kliendi- ja serveripoolseks liideseks lubab mõlemat komponenti üksteisest sõltumatult täiendada ja muuta. See aga annab omakorda võimaluse ka loodud elementide taaskasutamiseks mõne uue projekti raames.

## Kokkuvõte

Bakalaureusetöö eesmärk oli luua mobiilirakendus kuulutuses oleva info automaatseks kalendrisse sisestamiseks. Tuvastatava info alla kuulub sündmuse pealkiri, toimumisaeg ja -koht. Töö raames keskendutakse eesti- ja ingliskeelsete kuulutuste töötlemisele, kuid toetatakse vähemal määral ka muid ladina tähestikul põhinevaid keeli. Eesmärgi saavutamiseks vaadeldi erinevaid optilise märgituvastuse ja loomuliku keele töötluse vahendeid. Olemasolevate lahenduste uurimisel selgus, et ühtegi eesti keelt toetavat mobiilirakendust ei leidunud, mis sel viisil kuulutuselt info tuvastamist võimaldaks.

Töö raames valmis Androidi rakendus koos veebipõhise rakendusliidesega. Loodud mobiilirakendus võimaldab kasutajal teha kuulutusest nutitelefoni kaameraga pilti või kasutada mõnda olemasolevat kuulutuse pilti. Kasutades optilise märgituvastuse töövahendit Google Mobile Vision tuvastatakse kõigepealt sisendiks saadud pildilt kuulutuse tekst. Seejärel saadetakse tuvastatud tekst edasiseks analüüsiks veebipõhisele rakendusliidesele. Loodud veebiliides tuvastab päringuga saabunud teksti keele ning eraldab sealt ürituse algus- ja lõppaja ning toimumiskoha. Loomuliku keele töötluseks kasutatakse eestikeelsete kuulutuste puhul Pythoni teeki EstNLTK, muul juhul teeki spaCy. Saadud tulemused tagastatakse päringu teinud mobiilirakendusele, mille põhjal pannakse kokku sobiv kalendri sissekanne. Sündmuse pealkirjaks määratakse alati kuulutuselt leitud kõrgeim tekst. Kuulutuse töötluse lõppedes avatakse kasutajale automaatselt kalendrirakendus koos uue sündmuse lisamise vaatega. Enne sündmuse salvestamist saab kasutaja üle kontrollida automaatselt täidetud pealkirja, toimumisaja ja -koha väljad ning vajadusel teha parandusi.

Testimise tulemused näitasid, et loodud rakendus saab üsna adekvaatselt hakkama lihtsamat tüüpi kuulutuste töötlemisega, aga jääb hätta keerulisema kujundusega kuulutustelt info tuvastusega. Rakenduse täpsuse hindamiseks mõõdeti, kui kaua läheb sündmuse kalendrisse sisestamiseks aega mobiilirakendust kasutades ning kui kaua käsitsi. Saadud tulemuste võrdlusel selgus, et katses osalenud isikud säästsid mobiilirakenduse abiga sündmuse sisestamisel keskmiselt 13,8 sekundit. Kui parandada ka suurema müraga kuulutuste tuvastamise täpsust, oleks loodud rakendus piisavalt töökindel, et seda võiks eelistada manuaalsele meetodile.

# Viidatud kirjandus


[1] Aldhaban F. Exploring the adoption of Smartphone technology: Literature review. *2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies*, 2012, pp. 2758–2770.

[2] Giachetti C, Marchi G. Evolution of firms' product strategy over the life cycle of technology-based industries: A case study of the global mobile phone industry, 1980-2009. *BUSINESS HISTORY*, 2010, pp. 1123-1150.

[2] Hasegwa A, TomiyaYamazumi, Hasegwa S, et al. Evaluating the Input of Characters using Software Keyboards in a Mobile Learning Environment: A Comparison between Software Touchpanel Devices and Hardware Keyboards. *Mobile and Ubiquitous Technology in Education 2012 IEEE Seventh International Conference on Wireless*, 2012, pp. 214–217.

[3] Berchmans D, Kumar SS. Optical character recognition: An overview and an insight. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014, pp. 1361–1365.

[4] Fujisawa H. A View on the Past and Future of Character and Document Recognition. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, pp. 3–7.

[5] Verma A, Arora S, Verma P. OCR-Optical Character Recognition. *7th International Conferernce on Recent Innovations in Science, Engineering and Management*, 2016, pp. 181–191.

[6] Google Translate's instant camera translation gets an upgrade. *Google*, 2019. https://blog.google/products/translate/google-translates-instant-camera-translation-gets-upgrade/ (02.12.2019)

[7] Park J, Kwon Y-B. An Embedded OCR: A Practical Case Study of Code Porting for a Mobile Platform. *2009 Chinese Conference on Pattern Recognition*, 2009, pp. 1–5.

[8] Mobile Document Capture and Real-Time Recognition SDK - ABBYY. https://www.abbyy.com/en-eu/mobile-capture-sdk/ (02.12.2019)

[9] Tesseract OCR: Mobile. *GitHub*. https://github.com/tesseract-ocr/tesseract/wiki/User-Projects-%E2%80%93-3rdParty#mobile (02.12.2019)

[10] Mobile Vision. *Google Developers*. https://developers.google.com/vision (02.12.2019)

[11] ML Kit | Google Developers. https://developers.google.com/ml-kit/vision (02.12.2019)

[12] Hutchins WJ. The Georgetown-IBM Experiment Demonstrated in January 1954. *AMTA*, 2004. DOI: 10.1007/978-3-540-30194-3_12.

[13] Nadkarni PM, Ohno-Machado L, Chapman WW. Natural language processing: an introduction. *J Am Med Inform Assoc,* 2011, vol. 18, pp. 544–551.

[14] Jiang J. Information Extraction from Text. Mining Text Data. Boston, MA: Springer US. 2012.

[15] Albared M, Gallofré Ocaña M, Ghareb A, et al. Recent Progress of Named Entity Recognition over the Most Popular Datasets. *2019 First International Conference of Intelligent Computing and Engineering (ICOICE),* 2019, pp. 1–9.

[16] Estnltk – Open source tools for Estonian natural language processing — estnltk 1.4.1 documentation. https://estnltk.github.io/estnltk/1.4.1/index.html (14.04.2020)

[17] TextBlob: Simplified Text Processing — TextBlob 0.15.2 documentation. https://textblob.readthedocs.io/en/dev/index.html (14.04.2020)

[18] spaCy · Industrial-strength Natural Language Processing in Python. https://spacy.io/ (14.04.2020)

[19] Stanford CoreNLP – Natural language software | Stanford CoreNLP. https://stanfordnlp.github.io/CoreNLP/index.html (14.04.2020)

[20] Natural Language | Google Cloud. https://cloud.google.com/natural-language (14.04.2020)

[21] Snap Event - Apps on Google Play. https://play.google.com/store/apps/details?id=io.github.arjunkrishnababu96.snapevent&hl=en (20.04.2020)

[22] Google Lens - Apps on Google Play. https://play.google.com/store/apps/details?id=com.google.ar.lens&hl=en (20.04.2020)

[24] calendar-event-from-event-poster-app. *GitHub*. https://github.com/ckitama/calendar-event-from-event-poster-app (07.05.2020)

[25] calendar-event-from-event-poster-server. *GitHub*. https://github.com/ckitama/calendar-event-from-event-poster-server (07.05.2020)

[26] Google Calendar - Apps on Google Play. https://play.google.com/store/apps/details?id=com.google.android.calendar&hl=en (07.05.2020)

[23] Rahman M. How to find the Android Version Distribution statistics in Android Studio. *xda-developers,* 2020. https://www.xda-developers.com/android-version-distribution-statistics-android-studio/ (20.04.2020)

[24] API Code & Client Generator | Swagger Codegen | Swagger. https://swagger.io/tools/swagger-codegen/ (13.04.2020)

[25] OpenAPI Specification | Swagger. https://swagger.io/specification/ (13.04.2020)

[26] What is REST?. https://www.restapitutorial.com/lessons/whatisrest.html (13.04.2020)

[27] Named entity recognition — estnltk 1.4.1 documentation. https://estnltk.github.io/estnltk/1.4.1/tutorials/ner.html (13.04.2020)

[28] Working with text, Temporal expression (TIMEX) tagging — estnltk 1.4.1 documentation. https://estnltk.github.io/estnltk/1.4.1/tutorials/text.html?highlight=timex#temporal-expression-timex-tagging (13.04.2020)

[29] Annotation Specifications, Named Entity Recognition · spaCy API Documentation. https://spacy.io/api/annotation#named-entities (13.04.2020)

[30] Tutorial: Quickstart, Translation and Language Detection — TextBlob 0.15.2 documentation. https://textblob.readthedocs.io/en/dev/quickstart.html#translation-and-language-detection (13.04.2020)

[31] Why Docker? | Docker. https://www.docker.com/why-docker (28.04.2020)

[32] Training spaCy's Statistical Models · spaCy Usage Documentation. https://spacy.io/usage/training (27.04.2020)

[33] Rahvaarv rahvuse järgi, 1. jaanuar, aasta - Eesti Statistika. https://www.stat.ee/34267 (27.04.2020)

# Lisad

## I. REST-põhise rakendusliidese spetsifikatsioon

Loodud veebipõhise rakendusliidese spetsifikatsioon YAML-formaadis, mis järgib OpenAPI 3 standardit.

```
openapi: 3.0.1
info:
  title: NLP API
  description: API for calling various NLP methods
  version: 1.0.0
servers:
- url: /
tags:
- name: estnlp
  description: NLP tools for Estonian
- name: genericnlp
  description: NLP tools for other input languages
paths:
  /:
    get:
      operationId: root_get
      responses:
        "200":
          description: OK
          content:
            text/plain:
              schema:
                type: string
                example: Welcome to the NLP API
                x-content-type: text/plain
      x-openapi-router-controller: swagger_server.controllers.default_controller
  /autonlp/ner:
    post:
      tags:
      - autonlp
      summary: Convenience endpoint that delegates work to either estnlp or
genericnlp
        tools based on input language.
      operationId: auto_ner
      requestBody:
        description: Input text in any language
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TextDto'
        required: true
      responses:
        "200":
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/AutoNerResultDto'
        default:
          description: Invalid Request
          content: {}
      x-codegen-request-body-name: body
      x-openapi-router-controller: swagger_server.controllers.autonlp_controller
  /estnlp/ner:
    post:
      tags:
```

```
      - estnlp
      summary: Helps with named entity recognition using estnltk. Extended to
also
        include temporal info.
      operationId: est_ner
      requestBody:
        description: Input text in Estonian
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TextDto'
        required: true
      responses:
        "200":
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/EstNerResultDto'
        default:
          description: Invalid Request
          content: {}
      x-codegen-request-body-name: body
      x-openapi-router-controller: swagger_server.controllers.estnlp_controller
  /genericnlp/ner:
    post:
      tags:
      - genericnlp
      summary: Helps with named entity recognition using spaCy. Additionaly uses
TextBlob
        to detect input text's language and translate it to English if necessary
(as
        spaCy works best in English).
      operationId: generic_ner
      requestBody:
        description: Input text in any language
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TextDto'
        required: true
      responses:
        "200":
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/GenericNerResultDto'
        default:
          description: Invalid Request
          content: {}
      x-codegen-request-body-name: body
      x-openapi-router-controller:
swagger_server.controllers.genericnlp_controller
components:
  schemas:
    TextDto:
      type: array
      description: Can be an array of words, phrases, sentences etc.
      items:
        type: string
    AutoNerResultDto:
      type: object
      properties:
        per:
          type: array
          items:
            type: string
```

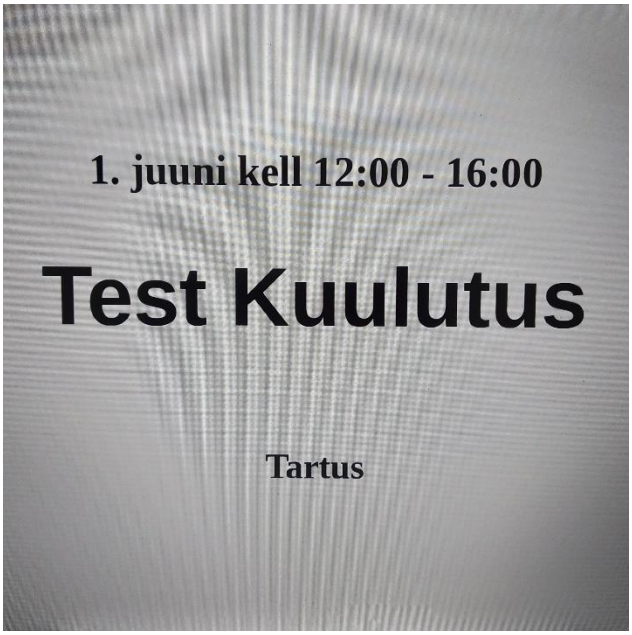
```
      org:
        type: array
        items:
          type: string
      loc:
        type: array
        items:
          type: string
      start:
        type: string
        description: Starting time in ISO 8601 format
      end:
        type: string
        description: Ending time in ISO 8601 format
    example:
      loc:
      - loc
      - loc
      org:
      - org
      - org
      start: start
      end: end
      per:
      - per
      - per
  EstNerResultDto:
    type: object
    properties:
      per:
        type: array
        items:
          type: string
      org:
        type: array
        items:
          type: string
      loc:
        type: array
        items:
          type: string
      start:
        type: string
        description: Starting time in ISO 8601 format
      end:
        type: string
        description: Ending time in ISO 8601 format
    example:
      loc:
      - loc
      - loc
      org:
      - org
      - org
      start: start
      end: end
      per:
      - per
      - per
  GenericNerResultDto:
    type: object
    properties:
      person:
        type: array
        items:
          type: string
      norp:
        type: array
```

```
      items:
        type: string
    fac:
      type: array
      items:
        type: string
    org:
      type: array
      items:
        type: string
    gpe:
      type: array
      items:
        type: string
    loc:
      type: array
      items:
        type: string
    product:
      type: array
      items:
        type: string
    event:
      type: array
      items:
        type: string
    work_of_art:
      type: array
      items:
        type: string
    law:
      type: array
      items:
        type: string
    language:
      type: array
      items:
        type: string
    date:
      type: array
      items:
        type: string
    time:
      type: array
      items:
        type: string
    percent:
      type: array
      items:
        type: string
    money:
      type: array
      items:
        type: string
    quantity:
      type: array
      items:
        type: string
    ordinal:
      type: array
      items:
        type: string
    cardinal:
      type: array
      items:
        type: string
    start:
      type: string
```

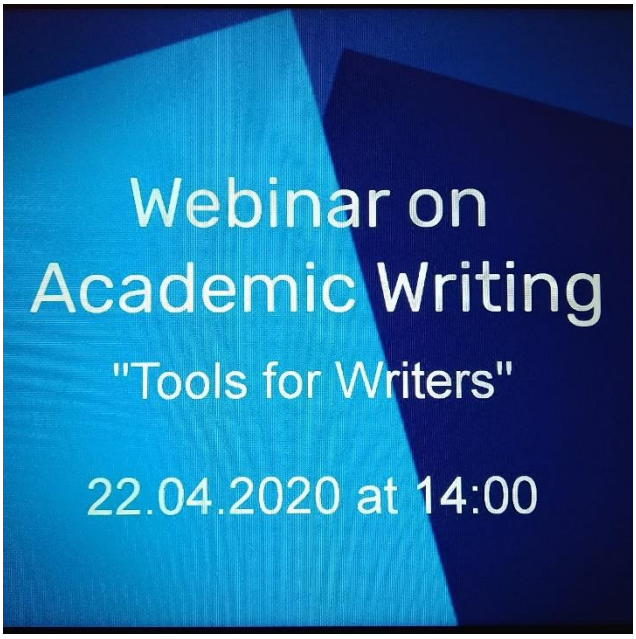
```
      description: Starting time in ISO 8601 format
    end:
      type: string
      description: Ending time in ISO 8601 format
  example:
    date:
    - date
    - date
    loc:
    - loc
    - loc
    product:
    - product
    - product
    law:
    - law
    - law
    quantity:
    - quantity
    - quantity
    org:
    - org
    - org
    start: start
    fac:
    - fac
    - fac
    language:
    - language
    - language
    work_of_art:
    - work_of_art
    - work_of_art
    cardinal:
    - cardinal
    - cardinal
    norp:
    - norp
    - norp
    percent:
    - percent
    - percent
    money:
    - money
    - money
    person:
    - person
    - person
    end: end
    time:
    - time
    - time
    event:
    - event
    - event
    gpe:
    - gpe
    - gpe
    ordinal:
    - ordinal
    - ordinal
```

## II. Testimiseks kasutatud kuulutused

Testimiseks kasutati viite kuulutust. Esimene neist põhineb käsitsi loodud mustvalgel tekstidokumendil, ülejäänud pärinevad interneti-portaalidest piletilevi.ee ja facebook.com.

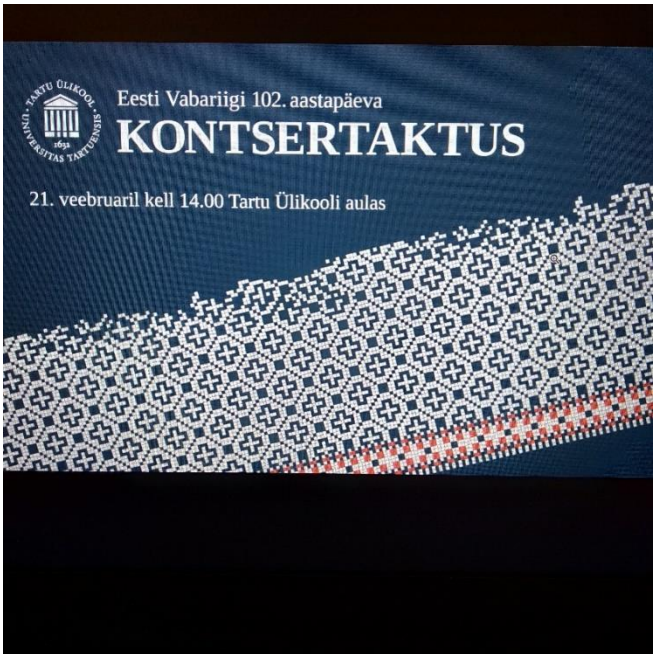



Joonis 8. Testvalimi esimene kuulutus.




Joonis 9. Testvalimi teine kuulutus[2].

[2] https://www.facebook.com/events/535912840445713/ (28.04.2020)




Joonis 10. Testvalimi kolmas kuulutus[3].


Joonis 11. Testvalimi neljas kuulutus[4].


Joonis 12. Testvalimi viies kuulutus[5].

[3] https://www.facebook.com/events/508632966740906/ (28.04.2020)
[4] https://www.piletilevi.ee/est/piletid/muusika/klassika/ac-dc-tribute-show-sumfooniaorkestriga-293611/ (28.04.2020)
[5] https://www.piletilevi.ee/est/piletid/kogupere/messid-ja-festivalid/kuressaare-merepaevad-2020-66280/ (28.04.2020)

# III. Testimise tulemused

Tabel 1. Mobiilirakenduse abil kalendrisse sündmuse lisamiseks kulunud aeg

| | 1. testija | 2. testija | 3. testija | 4. testija |
|---|---|---|---|---|
| **1. kuulutus** | 10 | 10 | 9 | 10 |
| **2. kuulutus** | 20 | 26 | 23 | 29 |
| **3. kuulutus** | 18 | 14 | 12 | 25 |
| **4. kuulutus** | 23 | 18 | 19 | 46 |
| **5. kuulutus** | 47 | 57 | 22 | 45 |

Tabel 2. Käsitsi sündmuse kalendrisse lisamiseks kulunud aeg.

| | 1. testija | 2. testija | 3. testija | 4. testija |
|---|---|---|---|---|
| **1. kuulutus** | 37 | 36 | 30 | 47 |
| **2. kuulutus** | 44 | 33 | 49 | 41 |
| **3. kuulutus** | 46 | 37 | 31 | 48 |
| **4. kuulutus** | 30 | 36 | 30 | 43 |
| **5. kuulutus** | 37 | 39 | 24 | 41 |

# IV. Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks**

Mina, Marion Laur,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Mobiilirakendus kuulutuses oleva info automaatseks kalendrisse sisestamiseks“, mille juhendaja on Sven Aller, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

*Marion Laur*

***08.05.2020***