TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Arvutiteaduse instituut

ITI40LT

Argo Käsper 134299

HTML 5 TEHNOLOOGIA EELISED JA PUUDUSED MOBIILIRAKENDUSTE ARENDAMISEL ADOBE PHONEGAP'I NÄITEL

Bakalaureusetöö

Juhendaja: Juhan-Peep Ernits
Phd
Dotsent
Kaasjuhendaja: Silver Kallas
BSc
Tegevjuht Aedes Web Solutions OÜ

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Argo Käsper

23.05.2015

Annotatsioon

Bakalaureustöö on koostatud uurimaks HTML5 tehnoloogia kasutamist mobiilirakenduste arendamisel. Sellel tehnoloogial kirjutati valmis üks hübriidmobiilirakendus Adobe PhoneGap raamistikus, et uurida ja võrrelda sellisel viisil arendatud rakendust platvormipõhise mobiilirakendusega. Seega valmistöö tulemusel lisaks ka Android'i mobiilirakendus, mis sisaldas ainult vaadeldava rakenduse põhifunktsioone: positsioneerimine, pildistamine ja rakendusesisesed teated.

Uurimusest selgus, et lihtsamate mobiilirakenduste jaoks on HTML5 tehnoloogia täiesti piisav ning lõpp tulemus näeb platvormipõhise mobiilirakendusega samane välja. Samuti on niiviisi arendada lihtsam, kuna ei ole vaja teada platvormipõhiseid programmeerimiskeeli ja ka rakenduse haldus on vähem tülikam, kuna hooldada on vaja vaid ühte koodi.

Uurimusest selgus veel ka see, et HTML5 tehnoloogiat kasutades ei saa seadme täisfunktsionaalsust alati täielikult ära kasutada, seega tuleb välja mõelda erilahendusi. Üks selline juhtum oli seoses positsioneerimisega, kus automaatne asukoha jälgimine ei olnud nii tõhus kui oma seadistustega koostatudspetsiifiline asukoha jälgimisfunktsioon. Sellest tuleb järeldada, et HTML5 tehnoloogia ja Adobe PhoneGap raamistikul on arenguruumi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 36 leheküljel, 7 peatükki, 9 joonist, 4 tabelit.

Abstract

Benefits and drawbacks of using HTML5 in mobile app development by Adobe PhoneGap example

The goal of the thesis was to investigate the possibilities of the application of web technologies in mobile app development. With the advent of HTML5 technology it is possible to access different APIs. A hybrid mobile app was developed using Adobe PhoneGap framework to study the possibilities of such technology and overcome the occurring drawbacks.

Alongside with the hybrid app a native Android app was also developed which had only the core functionalities of the hybrid app: location positioning, camera and in-app notifications. The purpose of developing the Android app was to investigate the in-app delays of calling the main functionalities.

The study showed that simple mobile apps lend themselves well to be developed in hybrid the way, because no in-depth knowledge in various platform specific programming languages is needed. Also one can develop this hybrid app onto multiple platforms with the same code. Thus the maintenance cost is much lower.

In the case, when an app needs to have access to more functionalities of the device, the difficulties started to occur. Based on this thesis one difficulty was with Geolocation API of Apache Cordova. Using the default configuration of this API did not function properly, thus the geolocation function call was modified in a way, that it enabled controlling the interval of each position request more strictly. This shows that hybrid apps tend to fall behind in performance compared to native apps, because the over abstraction of APIs still require a programmer to make extra code snippets due to some platform specific quirks.

Taking all this into consideration it is clear that the HTML5 technology still needs to evolve to make hybrid apps perform better, but the current status of HTML5 and other web technologies provide a reasonable quality of making simple hybrid apps effectively.

The thesis is in Estonian and contains 36 pages of text, 7 chapters, 9 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	Application programming interface, rakendusliides
HTML	HyperText Markup Language, veebilehtede märgendamise keel
CSS	Cascading Style Sheets, märgendite kujundamise keel
CLI	Command Line Interface, käsurida
HTTPS	HyperText Transfer Protocol Secure, turvaline hüperteksti edastusprotokoll
DOM	Document Object Model, märgendkeelte (HTML, XML, jne) märgendeid tõlgendatakse objektidena
OS	Operation System, operatsioonisüsteem
URL	Uniform Resource Locator, internetiaadress
AJAX	Asyncronous JavaScript and XML, asünkroonne andmete edastus
IP-aadress	Internet Protocol address, interneti aadress
GPS	Global Positioning System, satelliitnavigatsiooni süsteem
REST	Representational State Transfer, veebitehnoloogia arhitektuur andmete edastamiseks ja vastu võtmiseks
PHP	Hypertext Preprocessor, serveripoolne skriptimiskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö eesmärgiks on uurida hübriidmobiilirakenduse ja platvormipõhise mobiilirakenduse peamisi eeliseid ja puudusi ning ülesandeks valmis teha üks hübriidmobiilirakendus kasutades Adobe PhoneGap raamistikku. Paljudest hübriidarenduse raamistikest valisin välja just Adobe PhoneGap'i, sest see lubab koodi arenduseks valida ükskõik millise JavaScript raamistiku, samas teistel on see ette määratud – niisiis valisin Backbone JS[1] raamistiku, kuna olin seda varem kasutanud.

Hübriidmobiilirakenduse teen ma valmis Teooria OÜ e-õppe keskkonna teooria.ee õpetajate allsüsteemi sõidumoodili jaoks koostöös Aedes Web Solutions OÜ-ga. Seda rakendust arendatakse tiimis, kus on kolm liiget: arendaja, kujundaja ja projektijuht. Mina olen arendaja rollis ja mu ülesandeks on valmis kirjutada kogu funktsionaalsus ning see lõpuks kompileerida Androidi mobiilirakenduseks.

Töö võrdluse läbiviimiseks teen arendatavale rakendusele kõrvale ka Android'i platvormil ühe testrakenduse, mis sisaldab ainult võrreldavaid funktsionaalsusi kahe tehnoloogia vahel.

Peamised funktsionaalsused mida oma töös uurin:

1. Asukoha positsioneerimine,
2. Pildistamine,
3. Rakendusesisesed teavitused

[1] Backbone JS raamistikul on MVC struktuur, kus on mudelid ja kollektsioonid, mis hoiavad andmeid ja vaated, mis esitavad mudelite ja kollektsioonide andmeid. Kasutaja tegevuse jälgimine toimub vaadetes läbi JavasScript sündmuste (tõlgitud ingl. k. *events*). [20]

Põhiliseks mõõdikuks on ajakulu nimetatud funktsionaalsuste välja-kutsumiseks, kuna on levinud arvamus, et hübriidid on palju aeglasemad kui platvormipõhised. Samuti proovin anda võimalikult objektiivse hinnangu kummaski tehnoloogias arendamise kohta.

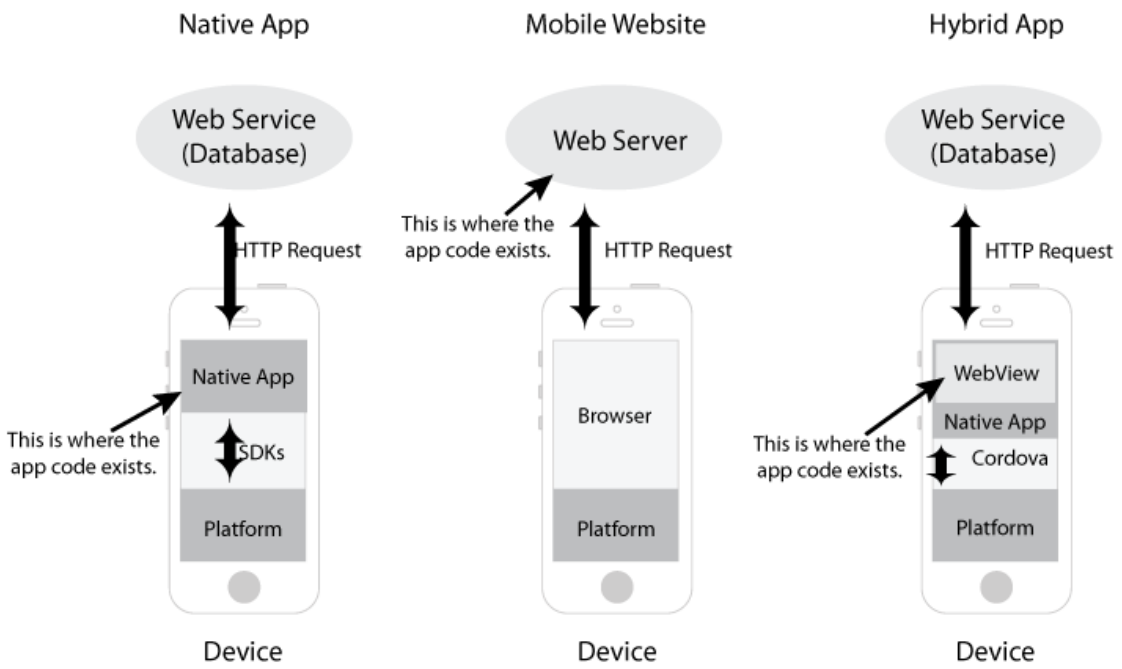
Kahjuks ei saa ma teostada mõõtmisi rakenduse operatiivmälu kasutuse kohta, kuna arendatavale hübriidrakendusele on seatud nõuded[1], mille ma pean täitma, aga Android'i testrakendusel ei näe ma vajadust neid kõiki nõudeid ära teha, et bakalaureusetöö eesmärki täita.

Lisaks võrreldavatele arendustehnoloogiatele on olemas ka kolmas, mis jääb kahe nimetatute vahele. Kolm tehnoloogiat on [1]:

1. **Platvormipõhine,**
2. Mobiilne veebileht,
3. **Hübriidmobiilirakendus**

Mobiilne veebileht on hübriidmobiilirakenduste eelkäija ja sisuliselt on need kaks sarnased, kuid erinevusega, et hübriidrakendusel asub kood kasutatavas seadmes, veebilehel aga serveris. Sellest tulenevalt on hübriidrakenduse sisemine arhitektuur siiski väga erinev ja sarnaneb pigem platvormipõhisele. Seega on leitud, et platvormipõhise ja mobiilise veebilehe tehnoloogiate kokku panemisel saab luua hübriidmobiilirakendusi, mis töötavad sama koodiga erinevatel platvormidel sama moodi. Ja kuna see tehnoloogia on alles uus ja pole veel väga levinud, siis ma loodan, et saan oma bakalaureusetööga selle tehnoloogia kasutuskogemust teistega jagada ja soovitada.

[1] Osad nõuded on – 1) Rakendus peab sisaldama sõitude ja õpilaste listi ning kalendrit. 2) Rakenduse kasutamiseks peab sisse logima. 3)Rakendus peab olema kujundatud, et õpetajal oleks seda mugav kasutada.



Joonis 1 Platvormipõhine arhitektuur (vasakul), mobiilse veebilehe arhitektuur (keskel) ja hübriidmobiilirakenduse arhitektuur (vasakul) [1].

Vaadates 2016. aasta aprilli seisuga statistikat, kuidas kasutatavad platvormid turul jagunevad, siis tulemus on järgmine: Android(62%), iOS(28%), Windows Phone(4%) jm(6%) [2]. Sellest saab järeldada et platvormide mitmekesisust veel leidub. Sellepärast on Adobe PhoneGap'i hübriidrakenduse lähenemine jätkuvalt aktuaalne.

2 Mobiilirakenduste erinevad arendustehnoloogiad

Mobiilirakenduste põhilisi arendustehnoloogiaid on kolm, aga oma bakalaureusetöös keskendun vaid kahele – platvormipõhine- ja hübriidarendus.

Platvormipõhise ja hübriidrakenduse erinevuseks on see, et ühega saab rakendust arendada vaid kindlale platvormile, samas teine võimaldab samast koodist kompileerida rakendusi erinevatele platvormidele.

2.1 Platvormipõhised arendusviisid

Platvormipõhist arendusviisi nimetatakse arendamist lähtuvalt platvormi spetsiifikale. Näiteks Android platvormile arendatakse programmeerimiskeeles Java, aga iOS platvormile Objective-C's või Swift'is. Need keeled on süntaktiliselt liiga erinevad (Vt. Lisa 2 ja Lisa 3) ning samuti nende kompilaatorid, et toetada mõlemat nimetatud platvormi, mistõttu tuleb mobiilirakendus alati arendada vastavas programmeerimiskeeles – valmis tuleb kirjutada kaks erinevat koodi.

Platvormipõhise arendusviisi eeliseks on see, et kastutatud tehnoloogia[1] on täpselt sellele platvormile mõeldud. Seega õigesti kirjutatud koodis pole midagi üleliigset ega ka kohti kus viga võiks tekkida ilma vastava veahalduseta. See annab mobiilirakendusele juurde nii töökindlust kui ka kiirust, kuna platvormi ressursse kasutatakse efektiivselt. Ja alati saab seadme võimekust täielikult ära kasutada. Platvormipõhise arenduse kasuks räägib ka see, et arendajal on lihtsam, kuna koodi kirjutamine on otsene – nii saab ka vigu palju kiiremini üles leida.

Platvormipõhiste mobiilirakenduste arendamise põhiliseks puuduseks on vajadus arendada iga platvormi jaoks eraldi rakendus. See on kulukam ja nõuab rohkem

[1] Programmeerimiskeel, nt Java või Objective-C.

arendajaid erinevate oskustega. Sellise mobiilirakenduse hooldus on samuti kulukam, kuna ressursse tuleb kulutada mitme erineva platvormi koodi üle vaatamise peale.

2.2 Hübriidmobiilirakenduste arendamine

Hübriidmobiilirakendused töötavad mitmel platvormil sama koodi baasil ning selle arendamiseks kasutatakse HTML5, CSS ja JavaScript tehnoloogiaid. Tuntumad raamistikud, lisaks Adobe PhoneGap'ile on IONIC, Mobile Angular UI, Intel XDK, Appcelerator Titanium, Sencha Touch ja Kendo UI. Kõik need toetavad vaid kindlaid JavaScript'i raamistikke, mis on natuke pärssivaks teguriks – peab enne arendama asumist uusi asju selgeks õppima – aga samas on nende kohta saadaval väga hea dokumentatsioon. Adobe PhoneGap'i põhiline erinevus teistest on selles, et PhoneGap otseselt ei loo rakendust, vaid pakendab faile nii, et see oleks nagu mobiilirakendus. [3]

Hübriidmobiilirakenduse arendamine ei vaja platvormil töötava programmeerimiskeele tundmist ega ka vastavas keeles kirjutamist, kuna kasutatavad raamistikud teevad selle töö ise ära, kompileerides platvormipõhise mobiilirakenduse. Sisuliselt ongi hübriidmobiilirakendus veebibrauser, mille sees jookseb veebileht.

Hübriidrakenduste põhiliseks eeliseks on ühe koodiga loodavad erinevad platvormipõhised mobiilirakendused. See lihtsustab koodikirjutamist ja vähendab meeskonna suurust mis mõjutab otseselt sellise arenduse hinda – odavam arendus. Odavam arendus ei tähenda, et kvaliteedis peaks järgi andma, vaid lihtsalt nii arendades saab kulusid kokku hoida. [1]

Alati ei saa kõiki seadme funktsionaalsusi hübriidrakendusse lisada, kuna parasjagu ei pruugi raamistik seda veel toetada. Adobe PhoneGap'il on võimalus kirjutada pistikprogramme. Niimoodi võimaldatakse kasutada efektiivsemalt seadme funktsionaalsusi, mistõttu see probleem kunagi lahendamata ei jääks. [4]

Aga on palju tõsisem puudujääk, mis on efektiivsus. See mõjutab kõige rohkem kasutajakogemust, kuna mõnedes situatsioonides võib mobiilirakendus väga aeglaseks minna ja selle vastu ei saa midagi teha. Seda põhjustab tehnoloogia arhitektuur, mis hõlmab endas palju rohkem kihte kui platvormipõhine arhitektuur. Vaata kahe tehnoloogia arhitektuuri põhimõttelist võrdlust Joonis 1, lk. 13. Rohkemate kihtide

arvuga arhitektuur võib põhjustada ka vigade peitu jäämist, kuna need tekivad sügavamates kihtides, kus ei pruugi veel nii head veatöötlust olla tehtud. [1]

3 Apache Cordova (Adobe PhoneGap)

Hübriidmobiilirakenduse raamistikest ei ole Apache Cordova kõige populaarsem, kuid kindlasti arendajatele kõige meelepärasem. Lisaks mobiilirakenduse arendamisele veebitehnoloogiatega saab endiselt ka platvormipõhist arendust jätkata pistikprogramme (tõlgitud ingl. k. *plugin*) kirjutades. Kasutatavad veebitehnoloogiad on HTML5, CSS3 ja JavaScript ning pistikprogrammide jaoks platvormipõhised keeled nagu Java, Objective-C või C++. [4]

3.1 Ajalugu

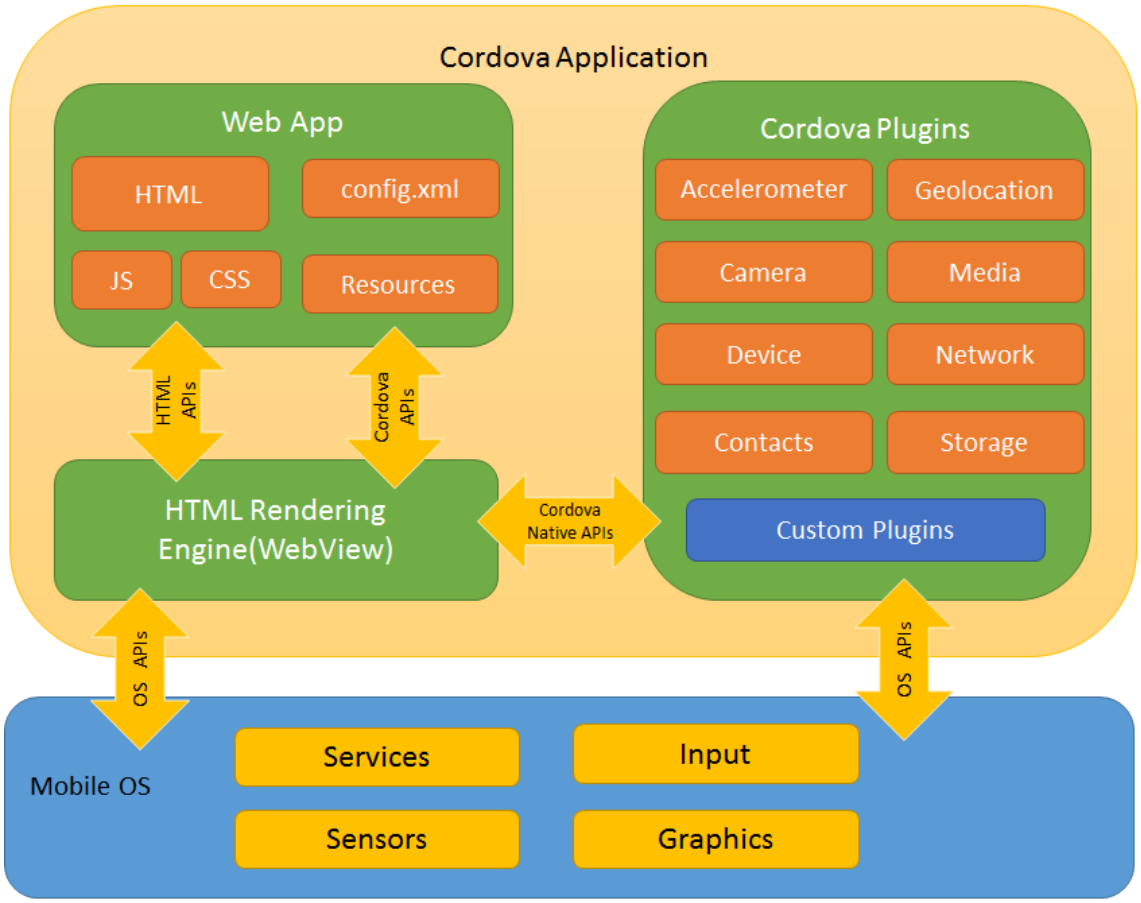
PhoneGap on avatud lähtekoodiga(*open-source*)[1] projekt, mis sai alguse firmast Nitobi. 2011. aastal ostis Adobe selle firma ära ning mõni aeg hiljem andis koodi Apache Software Foundation(ASF) hallata. ASF nimetas projekti ümber Apache Cordova'ks. Praegusega on Apache Cordova (PhoneGap) endiselt avatud lähtekoodiga ja pidev arendus sellega käib. Käesoleva töö esitamise hetkeks on uusim versioon, Cordova 6.0.0, mis lasti välja 28. jaanuaril 2016 [5]. [6]

3.2 Apache Cordova arhitektuur

Apache Cordova arhitektuur (Vt. Joonis 2, lk. 17) on põhimõttelt kahes osas: Cordova rakendus (tõlgitud ingl. k. *Cordova Application*) ja platvorm (tõlgitud ingl. k. *Mobile OS*). Cordova rakendus koosneb ise veel kolmest osast: veebirakendus (tõlgitud ingl. k. *Web*

[1] Avatud lähtekoodiga(*open-source*) tarkvara lähtekood ja dokumentatsioon on kõigile kasutajatele ja arendajatele vabalt kättesaadav nii tutvumiseks kui muutmiseks [21].

App), Cordova pistikprogrammid(tõlgitud ingl. k. *Cordova Plugins*) ja HTML tõlkemootor ehk veebivaade (tõlgitud ingl. k. HTML *Rendering Engine(WebView)*).



Joonis 2 Apache Cordova hübriidmobiilirakenduse arhitektuur [4]

3.2.1 Veebirakendus

Veebirakenduse osa on sisuliselt nagu veebileht ning kõik standardid, mis kehtivad veebilehe arenduses, kehtivad ka siin. See osa sisaldab kirjutatud koodi ning on aluseks terve rakenduse töötamiseks – selle eest vastutab *config.xml* fail. Vaata Lisa 4 Teooria mobiilirakenduse *config.xml* faili näidet.

config.xml hoiab endas vajaminevate koodide ja failide asukohti ja seadistusi ning selle põhjal kompileeritakse lõpuks mobiilirakendus.

3.2.2 Cordova pistikprogrammid

Cordova ja ka kolmandate osapoolte (tõlgitud ingl. k. *third-party*[1]) pistikprogrammid annavad võimaluse hübriidrakendusel kasutada seadme erinevaid komponente nii nagu seda teeb platvormipõhine mobiilirakendus. Nende abil saab seadme erinevaid funktsionaalsusi välja kutsuda JavaScript koodis. Cordova pistikprogrammide komplekti kutsutakse nimega *Core Plugins* [7], ning see sisaldab 20 pistikprogrammi, mis toetab 6 põhilist platvormi. Vaata pistikprogrammide näiteid Lisa 3 juurest. Lisaks neile on olemas ka kolmandate osapoolte pistikprogrammid.

3.2.3 HTML renderdusmootor ehk veebivaade

Veebivaade on kasutajaliides, mis ühendab kõik ülejäänud arhitektuuri komponendid. Kõige lihtsam oleks veebivaadet selgitada kui veebibrauserit, sest need kaks töötavad põhimõtteliselt ühte moodi – võtavad vastu HTML dokumendi ja renderdavad selle kasutajale nähtavaks ja interaktiivseks. Renderdamise (tõlgitud ingl. k. *rendering*) protsessi käigus kasutab brauser süsteemi ressursse ja komponente. Analoogselt toimuvad protsessid ka veebivaates. Veebivaade on Androidi hübriidrakenduses *Activity*[2]. [4]

[1] Third-party – selline tarkvara liigitus, mida saab kasutada koos mingi tarkvara või riistavaraga, kuid ei ole ametlikult seotud selle tarkvara tootja või kasutajaga. [22]

[2] Activity – Andoridi rakenduses täisekraan (mõnikord ka ujuvekraan) vaade, mida kasutaja näeb ja kasutada saab. [23]

3.3 Toetatavad platvormid

Apache Cordova toetab praegu seitset erinevat platvormi:

1. Android,
2. Blackberry 10,
3. iOS,
4. OS X,
5. Ubuntu,
6. Windows,
7. Windows Phone 8 ja Windows Phone 10.

Kõigile neile saab ühe ja sama koodiga kompileerida platvormipõhiseid rakendusi, mis oma olemuselt on – hübriidrakendused. Selle jaoks on vaja *config.xml* faili lisada iga platvormi seadistused, aga piisab ka globaalsetest seadistustest, mis sobivad kõikidele platvormidele, kui lihtsat rakendust arendada.

Kõiki funktsionaalsusi on raske igale platvormile sarnaselt rakendada, mistõttu osadele platvormidele arendamine on veel keerukas – vajaka-jäävate kohtade jaoks tuleb otsida *third-party* pistikprogramme või neid ise programmeerida. Androidil on olemas täisfunktsionaalsus [7] Cordova raamistikus, ning sellele platvormile arendades jääb hübriidmobiilirakendus ka kõige loomulikum. Cordova terviklik toetus Android'ile mõjutaski mind valima seda raamistikku.

4 HTML5

HTML5 sai alguse 2007. ning avalikustati 2012. aastal. Selles ajast peale on tehnoloogia veelgi edasi arenenud ja võimaldab veebitehnoloogiatega teha atraktiivseid veebilehti, mis sobivad arvutis ja mobiilis kasutamiseks. Samuti on sellest ajast peale arendatud ka mobiilirakendusi. Vahepealne areng on olnud nii suur, et tänapäeval saab luua veebitehnoloogiatega ka kvaliteetseid mobiilirakendusi.

Peamine uuendus HTML5's on see, et sellel on semantilisem süntaks. Nüüd on lihtsam HTML'i lugeda, kuna on sellised märgendid nagu *<header>, <footer>, <nav>* ja märgendite vahel saab luua ka loetavaid seoseid.

```
<input list="browsers">
<datalist id="browsers">
 <option value="Safari">
 <option value="Internet Explorer">
 <option value="Opera">
 <option value="Firefox">
</datalist>
```

Joonis 3 HTML struktuuris seoste loomine: *<input>* märgend hoiab endas *<datalist>* väärtusi läbi *list* atribuudi.

Lisaks on ka selliseid märgendeid, millega saab otse HTML struktuuri lisada multimeedia faile, näiteks *<video src="">, <audio src="">*

Samuti on HTML5 's nüüd võimalus küsida asukohta. Asukohta annab määrata erinevate viisidega: IP-aadressi, võrgumastide või GPS kiibi järgi ning HTML5 võimaldab seda otse küsida kasutades JavaScript'i.

HTML5 võimaldab andmete salvestamist, mis eelnevas versioonis oli veebibrauseri vahemälus (tõlgitud ingl. k. *cache*), aga nüüd saab kasutada WebSQL'i. See on mugav viis kasutajate andmete salvestamiseks ning koos HTML5 võrguühenduseta vahemälu (tõlgitud ingl. k. *offline cache*) abil on võimalik luua võrguühenduseta veebirakendusi. Võrguühendusete vahemällu ei salvestata kõike, vaid arendaja saab ise valida millised failid salvestatakse.

Lisatud on ka kaks erinevalt tüüpi andmehoidlat, mis asendavad küpsistes (tõlgitud ingl. k. *cookies*) hoitavaid kasutaja andmeid. Need on *sessionStorage* ja *localStorage* – nendest räägin oma töös täpsemalt hiljem. Need kaks mälu tüüpi võimaldavad andmeid hoida kohalikus seadmes struktureeritult. [8]

5 Teooria.ee mobiilirakendus

Teooria.ee on liiklusõppe keskkond, kus hetkeseisuga on liitunud juba üle 80 autokooli. Selline õppekeskkond toob kokku autokoolid, õpetajad ja õpilased ning hõlbustab nendevahelist infovahetust. Teooria.ee toetab praegu auditoorses õppes A, B, BE, C, CE, D, DE, ja T kategooriaid ja e-õppes A, B, BE, C ja CE kategooriaid.

Teooria.ee missiooniks on aidata kaasa hea liikluskultuuri ja üksteisega arvestava ühiskonna edendamisele Eestis.

Teooria.ee mobiilirakendus on mõeldud abistavaks töövahendiks sõiduõpetajatele. See hakkab asendama füüsilisi õpingukaarte ja sõidutundide jaoks vajaminevat kalendermärkmikku. Rakendus hakkab töötama Android'i tahvelarvutite peal.

5.1 Kasutusvaldkond ja põhifunktsionaalsused

Mobiilirakendus hakkab olema sõiduõpetaja töövahendiks füüsilise märkmiku asemel, kuna sisaldab endas õpilaste õpingukaarte, kalendrit sõitude jälgimist, hindamist ja teekonna salvestamist. Rakendus kasutab teooria.ee infosüsteemiga sama andmebaasi, seega kõik andmed on reaalajas sünkroniseeritud mobiilirakenduse ja veebiliidese vahel.

Rakendus on seotud teooria.ee infosüsteemis oleva õpetaja kontoga ja vajab enne kasutamist sisse-logimist. Andmevahetus rakenduse ja serveri vahel hakkab toimuma kasutades HTTPS protokolli.

Peamised funktsionaalsed nõuded mobiilirakendusel on:

1. Õpetajal kalendrisse vabade sõiduaegade üles märkimine, millele tema õpilased saavad veebikeskkonnas registreerida;

2. Registreeritud sõitude tühistamine õpetaja poolt;
3. Registreeritud sõitude sõitmine ning sõitude positsioneerimine;
4. Pärast sõitmist sõidu kinnitamine koos õpetaja ja õpilase allkirjastamisega;
5. Möödunud sõitude listi kuvamine ja nende info lugemine;
6. Õpilaste listi kuvamine ja nende info lugemine;
7. Õpilasest pildi tegemine ja selle lisamine tema õpingukaardile;
8. Õpetaja saab õpilasele uue sõidu registreerida.

5.2 Kasutatud tehnoloogiad, raamistikud ja APId

Kogu rakenduse arendan veebitehnoloogiatega, milleks on HTML5, CSS3 ja JavaScript. Rakenduse panin serveriga suhtlema läbi REST API. Selle jaoks kasutasin PHP'd.

Mobiilirakenduse arendamiseks kasutan JavaScript Backbone.JS raamistikku. Ja et valitud raamistikku kõik failid ja nende omavahelised sõltuvused mugavalt kokku viia, kasutan RequireJS raamistikku.

Adobe PhoneGap API'dest kasutan *Geolocation*, *Camera*, *Dialogs*, *Splashscreen*, *Vibration* ja *Whitelist* pistikprogramme ja *third-party* pistikprogrammidest kasutan *Insomia.*

5.2.1 REST API

Rakenduse jaoks oli lisaks vaja välja arendada vastav REST API, mis ühendaks andmebaasi rakendusega. Kogu andmevahetuse funktsionaalsus sai tehtud selle kaudu – sisse-logimine, sõitude listi kuvamine, õpilaste listi kuvamine, sõidu ja õpilase detailandmete kuvamine, kalendri jaoks kalendrisündmuste märkimine ja muutmine ja muude andmete lisamine ning lugemine.

5.2.2 Veebitehnoloogiad

Mobiilirakendustes on peaaegu alati vaja muutmälus hoida kasutaja hiljutisi toiminguid, et oleks infot kergem ja kiirem hiljem kätte saada. Selleks kasutan HTML5 *data*-atribuuti. *Data*- atribuut on HTML struktuuris globaalne, mis tähendab et tema väärtust

saab JavaScript'iga igas skoobis[1] küsida. See on hea ja mugav, aga kui tegemist on dünaamilise *data-* muutujaga, siis pärast *document*'i[2] värskendamist[3] on väärtus kadunud.

Veebibrauserid *data-* atribuute ei loe, seega võib nendele ohutult salvestada andmebaasist tulnud andmeid, et ei peaks liiga tihti uusi päringuid tegema. Siiski peab arvestama, et DOM'is on neid näha. [9]

data- atribuudi väärtust saab JavaScript'is kahte moodi küsida:

```
…
<div id="DataVal" data-something="value"></div>
<script>
  (function() {
    // esimene viis küsides otse atribuuti väärtust, väljastab „value“:
    $('#DataVal').attr('data-something');

    // teine viis küsides otse läbi JQuery raamistiku data() meetodi,
    // väljastab „value“:
    $('#DataVal').data('something');

  })(JQuery);
</script>
…
```

Joonis 4 *data-* attribuudi väärtuse küsimine JavaScript'is

Manipuleerides DOM *data* objekti JQuery data() meetodiga, ei teki üldse *data-* atribuuti, vaid see on otse DOM'i küljes.

---

[1] JavaScript'i skoop on muutujate, objectide ja functisoonide hulk, millele on ligipääs. JavaScipt on funktsioonilise skoobiga ehk skoop muutub funktsioonide sees. [24]

[2] *document* on globaalne objekt, mis hoiab endas hetkel kuvatavat HTML'i

[3] Tõlgitud ingl. k. *refresh*, brauseris olev HTML'i leht uuendatakse.

Veel kasutan kahte tüüpi andmete hoidlaid, mida veebitehnoloogia toetab: *SessionStorage*[1] ja *LocalStorage*[2]. Neid vajan selleks et andmeid kauem hoida, näiteks sisse loginud õpetaja andmeid ja seadistusi ning sõitudel kogutud asukohti. Aktiivse sõidu[3] korral hoian seal täpsemaid sõidu detaile.

SessionStorage on võetud arendatavas mobiilirakenduses kasutusele kui ühe sessiooni andmete mäluna. See hakkab hoidma sisse loginud õpetaja andmeid, et säästa päringute arvu andmebaasi, iga kord kui vaja mõnda õpetaja kasutajaga seotud infot küsida. Samuti salvestan sinna ka aktiivse sõidu andmeid samal põhjusel. *SessionStorage* poole saab pöörduda käsuga:

```
window.sessionStorage.getItem('itemName');
// või
window.sessionStorage.keyName;
```

ja salvestada saab käsuga:

```
window.sessionStorage.setItem('itemName', 'itemValue');
// või
window.sessionStorage.keyName = 'itemValue';
```

Analoogselt käib ka *LocalStorage* andmete käitlemine. Mõlemad mälutüübid on objektid, mis hoiavad andmeid tekstitüübina nimi-väärtuse paarina. Seega kui vaja teisi muutujatüüpe, siis tuleb need pärast mälust küsimist vastavasse tüübi konverteerida. [10]

LocalStorage's hakkan hoidma positsioneerimisel kogutud koordinaate ja ebaõnnestunud salvestamiste sõitude objekte.

---

[1] *SessionStorage* – hoiab sinna salvestatud andmeid ühe sessiooni ajaks ehk salvestatud andmed kaovad kui rakendus kinni panna või sellest välja logida. Kuulub *window* objekti alla.

[2] *LocalStorage* – hoiab sinna salvestatud andmeid määramatuks ajaks. Kui rakendus sulgeda või sellest välja logida, jäävad kõik andmed alles ja on kasutuskõlblikud. Kuulub *window* objekti alla.

[3] Aktiivne sõit on selline sõit, mis on parasjagu sõitmisel ja millele kogutakse asukoha andmeid.

5.2.3 Tähtsamad JavaScript raamistikud

5.2.3.1 Backbone.JS

Backbone.JS raamistiku abil saan ma tagada arendatavale koodile korraliku struktuuri. Struktuur hakkab olema MVC baasil. Lisas 4 on ära toodud failide struktuur teooria.ee mobiilirakenduse näitel. Kõige tähtsam fail selles raamistikus on *app.js* või *main.js*, mis on terve rakenduse käivitumise stardipakuks. Lisas 6 on *app.js* näide teooria.ee mobiilirakenduse põhjal.

Järgmise failina Backbone.JS raamistiku elu tsüklis kutsutakse välja *router.js*, mis vastutab erinevate vaadete eest. Lisas 7 on Teooria mobiilirakenduse koodilõik *router.js* failist. See fail vastutab erinevate URL'ide välja kutsumises ja need on siinmõistes päris URL'i ankrud ehk tekstid, mis on pärast „#" märki. Põhimõtteliselt selles raamistikus ei ole kordagi välja kutsutud lehekülje uuendamist, vaid uue sisu loomisel käib kõik läbi AJAX[1] päringute. Selles failis luuakse vaated vastavalt välja kutsutud URL'ile. [11]

Vaade formuleeritakse ette antud HTML failist, mis on isoleeritud teistest. Seega kõik JavaScript sündmused ja muud vaate tegevused toimuvad ainult seal ning pole kätte saadavad mujalt. Vaadetele saab luua alamvaateid ning siis pääsevad alam- ja ülemvaade (tõlgitud ingl. k. *subview and parent view*) üksteise muutujatele. objektidele ja funktsioonidele ligi. [12]

5.2.3.2 RequireJS

RequireJS raamistik võimaldab efektiivselt importida erinevaid JavaScript faile, mida koodi kirjutamisel vaja võib minna. RequireJS juures on väga hea see, et ta seob kõik JavaScript failid üheks ning ebakõlade korral teavitab koheselt – ehk kui kaks JavaScript faili miskipärast kokku ei sobi, siis RequireJS annab selles märku, vastupidiselt tavalise impordiga. Samuti annab RequireJS võimaluse kõikidest lisatud JavaScript failidest kokku panna üks suur ja optimeeritud JavaScripti fail – seda võimalust ma oma töös ei

[1] Ajax'iga saab tagaplaanil JavaScripti abil serverisse andmeid saata ja vastu võtta.

kasutanud, kuna rakendus on endiselt testperioodis ja kõikide failide sisu lugemine on vajalik. [13]

5.2.4 APId

Kasutusvaldkond ja põhifunktsionaalsusedPeatükis 5.1, lk 21 välja toodud funktsionaalsused saan ma täita vaid pistikprogrammide API'de abil, sest alguses ei sisalda Apache Cordova'ga hübriidrakenduse kompileerimine ühtegi seadme funktsionaalsust. Selle tõttu tuleb nad kõik lisada *config.xml* faili <*plugin*> märgendiga. Lisas 5 on toodud vastavad näited. Lisatavateks API'deks on:

- Whitelist (Core API)
- Geolocation (Core API)
- Camera (Core API)
- Dialogs (Core API)
- Splashscreen (Core API)
- Vibration (Core API)
- Insomnia (Third-party API)

Järgnevalt kirjeldan lühidalt, mida iga pistikprogramm arendatavas hübriidmobiilirakenduses tegema hakkab.

Whitelist vastutab rakendusse sisse tulevate ühenduste piiramist. Piiranguid saab seada domeenidele (nt. www.example.com), protokollidele (nt. http:// või ftp://) ja portidele (nt. 80, 443). Arendatavas rakenduses on piirang seatud nii, et lubatud on kõik ühendused domeenilt www.teooria.ee, mis tõstab turvataset, kuna IP aadressid ei saa niisama lihtsalt oma ühendusi luua. [14]

Enamus järgnevaid pistikprogramme saab otse JavaScriptis välja kutsuda *navigator* objektist.

Geolocation tegeleb seadme GPS mooduliga ja võrguga. Seda läheb vaja selleks, et jälgida sõitude marsruute. Põhilised meetodid, mida rakendus kasutab on *getCurrentPosition()* ja *watchPosition()*. [15] Arendatavas hübriidmobiilirakenduses kasutan *watchPosition()* meetodit, kuna see on täpsem, sest otsib asukohta kauem – kogub positsiooni kohta andmeid rohkematelt mastidelt ja satelliitidelt – ja see eest ka täpsemalt. [16]

Camera pistikprogramm hakkab käivitama seadme pildistamisrežiimi. Seda läheb vaja kui õpilasele on vaja õpingukaardile pilti ning õpilane lubab õpetajal seda teha. Pildistamisfunktsiooni saab välja kutsuda *navigator.camera.getPicture()* meetodiga.

Dialogs pistikprogrammi kasutan rakendusesiseste teavituste jaoks. Nt. kui on rakendusest väljumine, siis küsitakse kasutajalt enne üle, kas ta ikka soovib rakenduse sulgeda. Teavitusei on kahte tüüpi: *confirm* ja *alert*. *Confirm* on teavitus koos valiku variantidega ja järgnevate sündmustega, *alert* aga ainult teavitus ilma järgnevate sündmusteta.

Splashscreen on kasutusel selleks, et mobiilirakenduse käivitumisel näitaks Teooria logoga pilti. Kui seda ei kasutaks, näitaks käivitumisel valget või musta tausta.

Vibration API on selleks, et saada kasutada ja juhtida seadme värinat. Seda kasutan ma Teooria mobiilirakenduses kui on mingisugune hoiatus ehk *alert* tüüpi teavitus.

Insomnia on ainus kolmanda osapoole API, mida kasutan ja see on vajalik, et juhtida ekraani valgustust ja seadme unerežiimi minemist.

6 Android Java ja Adobe PhoneGap'i genereeritud Androidi rakenduse võrdlus

6.1 Android'i Java testrakendus

Võrdluseks hübriidmobiilirakendusele arendasin kõrvale ka platvormipõhise mobiilirakenduse. Rakendus sisaldab Teooria mobiilirakenduse põhifunktsionaalsusi: positsioneerimine, pildistamine ja rakendusesisesed teavitused. Nende kolme funktsionaalsusega sain võrrelda kas ja kui suur erinevus on kahe tehnoloogia vahel.

Android platvormipõhise testrakenduse koodi saab näha sellelt aadressilt: https://gitlab.com/aarx/final-thesis-app.git.

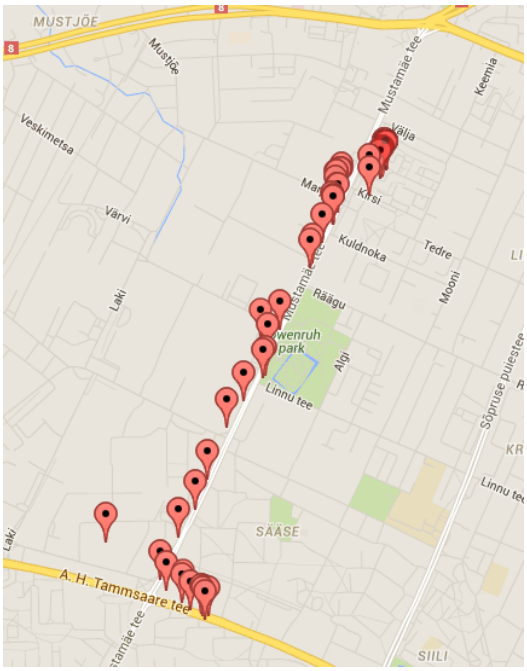
6.2 Positsioneerimine(*Geolocation*)

Asukoha positsioneerimisel ja punktide logi kogumisel sõitsin koos seadmega ühe lõigu, millel töötasid korraga mõlemad rakendused, hübriid ja platvormipõhine. Eelnevatest katsetustest selgus, et rakendused üksteist sellisel viisil töötamisel segama ei hakkaks. Mõlemad salvestasid sõidetud teekonda ja logisid.

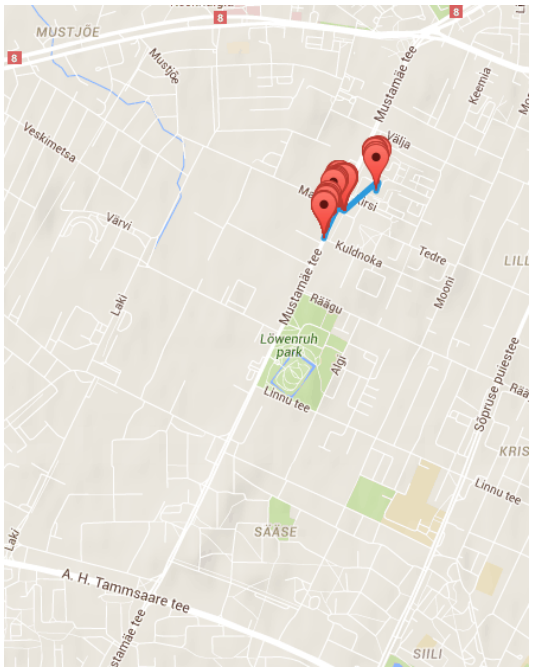
Hübriidrakenduse Geolocation API ei töötanud kahjuks nii nagu vaja oli. Enne katset sai testitud, kuidas rakendus asukohti kogub, kuid selgus et kui rakendus on taustal, siis ta enam ei positsioneeri. Seda põhjustab PhoneGap arhitektuur, kuna JavaScript'i koodi täitmine peatub kui rakendus panna taustale. [17]

Õnneks on sellele probleemile olemas lahendus kolmanda osapoole API'ga, milleks on *Cordova Background GeoLocation Plugin* [18]. See API jätkab JavaScript koodi käivitamist ka siis kui rakendus on mitteaktiivne ehk taustale viidud ja asukoha pärimismeetodid on Cordova Geolocation pistikprogrammiga suhteliselt sarnased.

Sellegipoolest sai hübriidrakenduse tulemustest välja lugeda andmeid, mida näeb Joonis 6, lk. 29. Andmed näitavad, et hübriidrakendus mõõtis asukohta palju kauem, kuid mitu korda täpsemalt. Hübriidrakendusel kulus asukoha leidmiseks keskmiselt 2540 ms, aga platvormipõhisel rakendusel vaid 43 ms. See võis tuleneda sellest, et platvormipõhisel rakendusel võeti asukohti ka vahemälust ning samal ajal otsiti taustal juba uut asukohta. Seega üsna raske oli platvormipõhisel rakendusel määrata asukoha otsimise algusaega ja see salvestada. Sellegipoolest oli tuntav erinevus ka täpsuses – hübriidrakendusega mõõdeti asukoht täpsemalt kui platvormipõhises. See seletab ka asjaolu, miks hübriidrakenduses asukoha otsimine kauem aega võttis.



Joonis 5 Android testrakenduse positsioneerimine



Joonis 6 Hübriidrakenduse positsioneerimine

Tabel 1 Positsioneerimise logi võrdlus

	Hübriidmobiilirakendus	**Platvormipõhine mobiilirakendus**
Kogutud andmete arv	98	99
Keskmine viivitus[1]	2540 ms	43 ms
Keskmine täpsus[2]	9 m	42 m

[1] Keskmine viivitus tähendab seda, et kõikide mõõdetud ajakuludest on võetud keskmine ajakulu e. viivitus.

[2] Keskmine täpsus tähendab positsioneerimisel seda, et kõikide salvestatud asukohtade objektidest on võetud täpsuse parameeter ja nendest on arvutatud keskmine. Täpsust mõõdetakse meetrites ja see näitab kaugust tegelikust asukohast.

6.3 Kaamera(*Camera*)

Kaamera funktsionaalsuse viivitust testisin 10 korda mõõtes aega nupule vajutusest (salvestasin algusaja) kuni kaamera tööle hakkamisega (salvestasin lõpu aja). Ajavahe saamiseks lahutasin need kaks aega omavahel. Hübriidmobiilirakendusel kasutasin ajahetke saamiseks *new Date()* enne ja pärast kaamera väljakutsumise meetodit.

```
this.startTime = (new Date()).getTime();
navigator.camera.getPicture(this.onSuccess,this.onError,window.cameraOptions
);
this.endTime = (new Date()).getTime();
```

Joonis 7 JavaScript'is kaamera väljakutsumise aja mõõtmine

Andorid'i platvormipõhisel rakendusel kasutasin ajahetke saamiseks *System.currentTimeMillis()* enne ja pärast kaamera väljakutsumise meetodit.

```
long startTime = System.currentTimeMillis();
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}
long endTime = System.currentTimeMillis();
```

Joonis 8 Javas kaamera väljakutsumine ja aja mõõtmine

Tabel 2, lk 31 on näha kogutud logiandmete tulemusi. Andmetest saab järeldada, et hübriidmobiilirakendusel kulub keskmiselt rohkem aega kaamera avamiseks Android platvormil. Suurim viivitus, 53 ms, salvestati esmasel kaamera käivitamisel. Järgnevad kaamera väljakutsumiseks vajalikud parameetrid võeti seadme vahemälust ja seetõttu oli viivitus ka väiksem. Platvormipõhisel Androidi rakendusel nii suurt erinevust viivitustes pole, kuna arhitektuuriliselt on Android'i rakendus ja kaamera tarkvara rohkem seotud. Viivitus oli keskmiselt 11ms ja varieerus 9 ms ja 12 ms vahel.

Tabel 2 Kaamera sisse lülitamise logi võrdlus

	Hübriidmobiilirakendus	**Platvormipõhine mobiilirakendus**
Kogutud andmete arv	10	10
Keskmine viivitus	27 ms	11 ms
Suurim viivitus	53 ms	12 ms
Väikseim viivitus	19 ms	9 ms

6.4 Teavitused(*Dialogs*)

Rakendusesiseste teavituste testid olid analoogselt sooritatud kaamera testidega. Kokku tegin 10 testi ja mõõtsin kulunud aega teavituse välja kutsumiseks. Algus ja lõpp ajamärgid salvestasin sama moodi nagu eelnevatel funktsionaalsustel, seega protseduur oli juba selge.

Siiski tulemustes oli üks suur ebakõla (Vt. Tabel 3, lk 32 ja Tabel 4, lk 33), Andorid'i platvormipõhisel rakendusel kulus *alert* teavituste välja kutsumiseks alati 10-20ms, kuid *confirm* teavituste jaoks keskmiselt ~1ms. Selline tulemus oli alati ja siis ma avastasingi seose, et Android'i seade jätab varem käivitatud protseduurid mällu ning kui kutsutakse samade parameetritega protseduure uuesti, võetakse see otse mälust, mitte ei arvutata uuesti kõiki vajaminevaid koodiridu. *alert* teavituse puhul täideti aga alati kõik vajaminevad koodiread (Vt. all olevatelt joonistelt Joonis 9, lk 31 ja Joonis 10, lk 32)

```
long startTime = System.currentTimeMillis();
CustomDialog dialog = new CustomDialog();
dialog.show(this.getFragmentManager(), "ConfirmFragment");
long endTime = System.currentTimeMillis();
```

Joonis 9 Confirm teavituse viivituse mõõtmine

```
long startTime = System.currentTimeMillis();
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle(R.string.dialog_confirm_title)
        .setMessage(R.string.dialog_alert_message)
        .setPositiveButton(R.string.alert_ok_message, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                //
            }
        });
builder.show();
long endTime = System.currentTimeMillis();
```

Joonis 10 Alert teavituse viivituse mõõtmine

Vaadates tulemusi Tabel 3, lk. 32 ei jää hübriidrakendus ka selles kriteeriumis sugugi alla.

Tabel 3 *Confirm* teavituse logi võrdlus

	Hübriidmobiilirakendus	**Platvormipõhine mobiilirakendus**
Kogutud andmete arv	10	10
Keskmine viivitus	4 ms	1 ms
Suurim viivitus	6 ms	14[1] ms
Väikseim viivitus	3 ms	0 ms

[1] Selline suur erinevus ülejäänutest tulenes sellest et see oli esmane vastava funktsiooni kasutamine. Kui korra oli teavituse objekti viide loodud, siis kasutati uuesti selle objekti loomisel eelneva objekti viidet, mistõttu ülejäänud viivitused olid väga väikesed.

Tabel 4 *Alert* teavituse logi võrdlus

	Hübriidmobiilirakendus	**Platvormipõhine mobiilirakendus**
Kogutud andmete arv	10	10
Keskmine viivitus	3 ms	15[1] ms
Suurim viivitus	7 ms	17 ms
Väikseim viivitus	2 ms	14 ms

Kogutud andmetest selgus, et erinevus kahe tehnoloogia vahel ei olegi väliselt nii märgatav testitud funktsionaalsuste põhjal. Tegevuste väljakutsumiseks kulub kummalgi rakendusel keskmiselt sama palju aega, umbes 10-30ms – selline ajaline viide ei ole kasutajale märgatav. Suurim erinevus erines positsioneerimisel, aga seda ka sellepärast et hübriidrakenduse oli Geolocation API seadistatud nii, et täpsus oleks suurim, aga platvormipõhisel rakendusel polnud seda seadistust tehtud.

Suurima viitelise erinevusega oli ka kaamera funktsiooni välja kutsumine, mis oli 2x kiirem platvormipõhisel rakendusel. Samuti oli märgata trendi, et funktsionaalsuse esmasel väljakutsumisel on rakendumisaeg pikem kui järgnevatel. See tuleneb platvormi enda mälu optimeerimisest ega pole seotud kahe tehnoloogia erinevusega.

[1] Erinev tulemus, kuna alati täideti funktsionaalsuse välja kutsumiseks kõik *alert* teavituse jaoks tarvis minevad koodiread

7 Kokkuvõte

Oma bakalaureusetöö teema sain ettevõttest Aedes Web Solutions OÜ, kus oli mulle antud ülesanne valmis arendada mobiilirakendus. Kuna ettevõte tegeleb igapäevaselt e-kaubandusega, siis oli mobiilirakenduse arendamine nende poolt esimene ning seepärast valiti ühisel kokkuleppel arendustehnoloogiaks veebitehnoloogiad ning hübriidmobiilirakendus, mitte platvormipõhine. Mobiilirakendust arendasin tiimis, kus olid arendaja, kujundaja ja projektijuht. Arendaja rolli täitsin mina. Mobiilirakendus kompileeriti Android'i platvormile ettevõtte Teooria OÜ tellimusel sõiduõpetajatele sõitude mugavamaks haldamiseks.

Oma bakalaureusetöös keskendusin põhiliselt HTML5 tehnoloogia mobiilirakenduse arendamisele. Sellele lisaks programmeerisin bakalaureusetöö raames võrdluseks kõrvale Android platvormipõhise testmobiilirakenduse, et tunda erinevust nii arendamisel kui ka rakenduse kasutamisel.

Oma töö esimeses osas uurisin taustainfot hübriid- ja platvormipõhise mobiilirakenduse kohta ning õppisin selgeks uusi raamistikke: hübriidi jaoks Backbone.JS ja Adobe PhoneGap ning platvormipõhise jaoks Android SDK. Samuti uurisin võimalikke erinevusi kahe tehnoloogia vahel – eeliseid ja puudujääke mõlema puhul.

Teine osa tööst oli kahe mobiilirakenduse valmis programmeerimine. Teooria mobiilirakenduse arendasin vastavalt nõuetele ja seetõttu oli eriti tähtis igasuguste vigade eemaldamine. Mõnede funktsionaalsuste lisamisel nagu näiteks positsioneerimine, oli vaja teha erilahendusi, et see korralikult töötaks. Kokkuvõttes hakkas rakendus korralikult tööle ja on praegu sõiduõpetajate käes testimises.

Android'i testrakenduse programmeerisin Android Studio IDE's mis aitas mugavalt ja kiirelt rakendust arendada ja kompileerida. Testrakendus sisaldas vaid kolme põhifunktsionaalsust, mis Teooria mobiilirakenduses olid: positsioneerimine, pildistamine ja rakendusesisesed teavitused. Lisaks tegin mõlemale rakendusele võimaluse koguda logiandmeid nimetatud põhifunktsionaalsuste kohta.

Kolmandaks osaks minu töös oli logiandmete võrdlus ja hinnang HTML5 tehnoloogias mobiilirakenduse arendamise kohta. Logiandmete kogumiseks kasutasin Samsung'i tahvelarvutit, mis töötas Android platvormil ning mille sain Mektory SmartLab keskkonnast laenutada.

Kogutud andmetest selgus, et erinevus kahe tehnoloogia vahel ei olegi väliselt nii märgatav testitud funktsionaalsuste põhjal. Tegevuste väljakutsumiseks kulub kummalgi rakendusel keskmiselt sama palju aega, umbes 10-30ms – selline ajaline viide ei ole kasutajale märgatav. Suurima viitelise erinevusega oli kaamera funktsiooni välja kutsumine, mis oli 2x kiirem platvormipõhisel rakendusel. Samuti oli märgata trendi, et funktsionaalsuse esmasel väljakutsumisel on rakendumisaeg pikem kui järgnevatel. See tuleneb platvormi enda mälu optimeerimisest ega pole seotud kahe tehnoloogia erinevusega.

HTML5 tehnoloogias arendamisel oli märgata puudujääke, mis olid eelneva taustauuringu tulemuse juba teada. Näiteks iga seadme funktsionaalsuse kasutamiseks on Adobe PhoneGap'i puhul vaja lisada pistikprogramme. Vaid nii saab välja kutsuda seadme erinevaid funktsionaalsusi. Ja kui on vaja erilahendusi, siis peab kas JavaScript'is lisa meetodeid koodi kirjutama või programmeerima ise pistikprogramme. Minu töö puhul piisas JavaScript'i muutmisest nii palju, et funktsionaalsus lõpuks tööle saada. Võimekamate mobiilirakenduste puhul on kindlasti vaja rohkem pistikprogrammi laadseid muudatusi koodis teha või kaaluda üldse juba platvormipõhise rakenduse arendamist.

Üks kõige suurem puudujääk PhoneGap raamistikus oli veel see, et rakenduse taustale ümber lülitamisel positsioneerimine lakkas töötamast. Selle lahenduseks on olemas kolmanda osapoole pistikprogrammid, nt. *Cordova Background GeoLocation Plugin*, mis jätkab positsioneerimist ka siis kui rakendus on taustprotsessile viidud.

Sellegipoolest julgen soovitada mobiilirakendusete arendamist hübriidina, kuna arendamine oli kerge ja ei nõudnud platvormialaseid teadmisi. Kuna see tehnoloogia on praegu ka väga kiiresti arenev, mida näitab asjaolu, et arenduse ajal uuenes Adobe PhoneGap päris mitme versiooni võrra edasi, siis on märgata, et hübriidmobiilirakendused on praegu see suund, kuhu liigutakse. Lisaks on arendamisele abiks väga suur hulk veebis olevatele materjalidele ja dokumentatsioonidele.

Edukalt veebitehnoloogiaid kasutades valmis programmeeritud Teooria hübriidmobiilirakendus peaks kasutusse minema 2016 aasta suve-/sügisperioodis, kui rakendus on testperioodi korralikult läbinud. Hiljem on plaanis sama rakendus kompileerida ka iOS süsteemidele ja võimalik et ka muudele platvormidele kui see tuleb vajalikuks.

Kasutatud kirjandus

[1] J. Wilken, „The Three Types of Mobile Experiences,“ DZone, 15 Aprill 2015. [Võrgumaterjal]. Available: https://dzone.com/articles/three-types-mobile-experiences. [Kasutatud 14 Mai 2016].

[2] „https://www.netmarketshare.com,“ Net Applications, Aprill 2016. [Võrgumaterjal]. Available: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1. [Kasutatud 13 Mai 2016].

[3] J. Raj, „The Top 7 Hybrid Mobile App Frameworks,“ Sitepoint, 7 November 2014. [Võrgumaterjal]. Available: http://www.sitepoint.com/top-7-hybrid-mobile-app-frameworks/. [Kasutatud 14 Mai 2016].

[4] Apache Cordova, „Apache Cordova Documentation,“ Apache Cordova, 12 Aprill 2016. [Võrgumaterjal]. Available: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. [Kasutatud 13 Mai 2016].

[5] S. Gill, „Cordova 6.0.0 Released!,“ Cordova, 28 Jaanuar 2016. [Võrgumaterjal]. Available: https://cordova.apache.org/news/2016/01/28/tools-release.html. [Kasutatud 13 Mai 2016].

[6] J. Jain, „Apache Cordova: Powerful Framework for Hybrid Mobile App Development,“ CodeProject, 13 Jaanuar 2016. [Võrgumaterjal]. Available: http://www.codeproject.com/Articles/1069661/Apache-Cordova-Powerful-Framework-for-Hybrid-Mobil. [Kasutatud 15 Mai 2016].

[7] Apache Cordova contributors, „Platform Support,“ Apache Cordova, 23 Aprill 2016. [Võrgumaterjal]. Available: https://cordova.apache.org/docs/en/latest/guide/support/index.html#core-plugin-apis. [Kasutatud 15 Mai 2016].

[8] F. Cimo, „Top 10 Major Advantages of HTML5,“ Web Code Geeks, 1 September 2015. [Võrgumaterjal]. Available: https://www.webcodegeeks.com/html5/top-10-major-advantages-html5/. [Kasutatud 19 Mai 2016].

[9] W3Schools contributors, „HTML data-* Attributes,“ W3Schools, [Võrgumaterjal]. Available: http://www.w3schools.com/tags/att_global_data.asp. [Kasutatud 18 Mai 2016].

[10] W3Schools, „HTML5 Local Storage,“ W3Schools, [Võrgumaterjal]. Available: http://www.w3schools.com/html/html5_webstorage.asp. [Kasutatud 15 Mai 2016].

[11] Backbone.JS, „Routing with URLs,“ Backbone.JS, [Võrgumaterjal]. Available: http://backbonejs.org/#Routing. [Kasutatud 18 Mai 2016].

[12] Backbone.JS, „View Rendering,“ BackBone.JS, [Võrgumaterjal]. Available: http://backbonejs.org/#View-rendering. [Kasutatud 18 Mai 2016].

[13] RequireJS, „HOW TO GET STARTED WITH REQUIREJS,“ RequireJS, [Võrgumaterjal]. Available: http://requirejs.org/docs/start.html. [Kasutatud 19 Mai 2016].

[14] Apache Cordova, „cordova-plugin-whitelist,“ Apache Cordova, 23 Aprill 2016. [Võrgumaterjal]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist/index.html. [Kasutatud 19 Mai 2016].

[15] Apache Cordova contributors, „cordova-plugin-geolocation,“ Apache Cordova, 23 Aprill 2016. [Võrgumaterjal]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-geolocation/index.html. [Kasutatud 19 Mai 2016].

[16] A. Gup, „How Accurate is HTML5 Geolocation, really? Part 2: Mobile Web,“ The Page Not Found Blog, 23 Juuni 2013. [Võrgumaterjal]. Available: http://www.andygup.net/how-accurate-is-html5-geolocation-really-part-2-mobile-web/. [Kasutatud 19 Mai 2016].

[17] J. Morony, „Background Geolocation with Phonegap Build,“ JoshMorony, 26 Märts 2014. [Võrgumaterjal]. Available: http://www.joshmorony.com/background-geolocation-with-phonegap-build/. [Kasutatud 23 Mai 2016].

[18] Cordova Background GeoLocation Plugin collaborators, „NPMJS,“ Transistor Software, Juuli 2015. [Võrgumaterjal]. Available: https://www.npmjs.com/package/cordova-plugin-background-geolocation. [Kasutatud 20 Mai 2016].

[19] C. Maunder ja D. Cunningham, „Apache Cordova: Powerful Framework for Hybrid Mobile App Development,“ CodeProject, 13 Jaanuar 2016. [Võrgumaterjal]. Available: http://www.codeproject.com/Articles/1069661/Apache-Cordova-Powerful-Framework-for-Hybrid-Mobil. [Kasutatud 13 Mai 2016].

[20] Backbone JS, „Models and Views,“ Backbone JS, [Võrgumaterjal]. Available: http://backbonejs.org/#Model-View-separation. [Kasutatud 15 Mai 2016].

[21] Wikipedia contributors, „Avatud lähtekoodiga tarkvara,“ Wikipedia, 27 Aprill 2013. [Võrgumaterjal]. Available: https://et.wikipedia.org/w/index.php?title=Avatud_l%C3%A4htekoodiga_tarkvara&oldid=3617494. [Kasutatud 14 Mai 2016].

[22] Wikipedia contributors, „Third-party software component,“ Wikipedia, 22 Aprill 2016. [Võrgumaterjal]. Available: https://en.wikipedia.org/w/index.php?title=Third-party_software_component&oldid=716577799. [Kasutatud 15 Mai 2016].

[23] Android Developers, „Activity,“ Android , [Võrgumaterjal]. Available: https://developer.android.com/reference/android/app/Activity.html. [Kasutatud 17 Mai 2016].

[24] W3Schools contributors, „JavaScript Scope,“ W3Schools, [Võrgumaterjal]. Available: http://www.w3schools.com/js/js_scope.asp. [Kasutatud 17 Mai 2016].

Lisa 1 – Apache Cordova platvormide toetus

	Android	**Blackberry10**	**iOS**	**Ubuntu**	**wp8 (Windows Phone 8)**	**windows (8.1, 10, Phone 8.1**
cordova CLI	+ Mac, Windows, Linux	+ Mac, Windows, Linux	+ Mac	+ Ubuntu	+ Windows	+
Embedded WebView	+	-	+	+	-	-
Plugin Interface	+	+	+	+	+	+
	Core Plugin APIs					
Accelerometer	+	+	+	+	+	+
BatteryStatus	+	+	+	-	+	+ Windows Phone 8.1
Camera	+	+	+	+	+	+
Capture	+	+	+	+	+	+
Compass	+	+	+ (3GS+)	+	+	+
Connection	+	+	+	+	+	+
Contacts	+	+	+	+	+	Osaliselt

	Android	**Blackberry10**	**iOS**	**Ubuntu**	**wp8 (Windows Phone 8)**	**windows (8.1, 10, Phone 8.1**
Device	+	+	+	+	+	+
Events	+	+	+	+	+	+
File	+	+	+	+	+	+
File Transfer	+	+ * Ei toeta *onprogress* või *abort* sündmust	+	-	+ * Ei toeta *onprogress* või *abort* sündmust	+ * Ei toeta *onprogress* või *abort* sündmust
Geolocation	+	+	+	+	+	+
Globalization	+	+	+	+	+	+
InAppBrowser	+	+	+	+	+	Kasutab *iframe*'i
Media	+	+	+	+	+	+
Notification	+	+	+	+	+	+
Splashscreen	+	+	+	+	+	+
Status Bar	+	-	+	-	+	+ Windows Phone 8.1
Storage	+	+	+	+	+ *localStorage* & *indexed DB*	+ *localStorage* & *indexed DB*
Vibration	+	+	+	-	+	+ Windows Phone 8.1

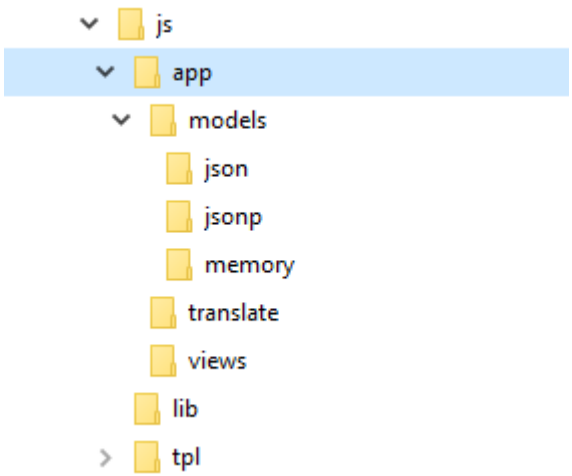
Lisa 2 – HelloWorld näide Objective-C keeles

```
#import <Foundation/Foundation.h>
int main (int argc, const char * argv[])
{
  NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
  NSLog (@"Hello, World!");
  [pool drain];
  return 0;
}
```

Lisa 3 – HelloWorld näide Java keeles

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

Lisa 4 – teooria.ee mobiilirakenduse failide struktuur



Lisa 5 – Teooria hübriidmobiilirakenduse *config.xml* fail

```
<?xml version='1.0' encoding='utf-8'?>
<widget            id="ee.teooria.mobiilirakendus"            version="1.0.0"
xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Teooria</name>
    <description>
        Teooria mobiilirakendus.
    </description>
    <author email="silver@aedes.ee" href="http://aedes.ee">
        Aedes Web Solutions
    </author>
    <content src="index.html" />
    <preference name="orientation" value="landscape" />
    <preference name="fullscreen" value="true" />
    <preference name="SplashScreen" value="screen" />
    <preference name="SplashScreenDelay" value="10000" />
    <plugin name="cordova-plugin-whitelist" spec="1" />
    <plugin name="App" value="org.apache.cordova.App"/>
    <plugin name="org.apache.cordova.dialogs" />
    <plugin name="org.apache.cordova.vibration" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <platform name="android">
        <allow-intent href="market:*" />
    </platform>
</widget>
```

Lisa 6 – teooria.ee mobiilirakenduse app.js fail

```
require.config({
    baseUrl: 'js/lib',
    paths: {
        app: '../app',
        tpl: '../tpl',
        trans: '../app/translate'
    },
    map: {
        '*': {
            'app/models/client': 'app/models/json/client',
            'app/models/teacher': 'app/models/json/teacher',
            'app/models/driving': 'app/models/json/driving',
            'app/models/apikey': 'app/models/json/apikey'
        }
    },
    shim: {
        'backbone': {
            deps: ['jquery', 'underscore'],
            exports: 'Backbone'
        },
        'underscore': {
            exports: '_'
        }
    }
});
define(function (require) {
    "use strict";
    var $                   = require('jquery'),
        Backbone            = require('backbone'),
        Router              = require('app/router'),
        LayoutView          = require('app/views/Layout'),
        Translate           = require('trans/translate');
   var app = {
```

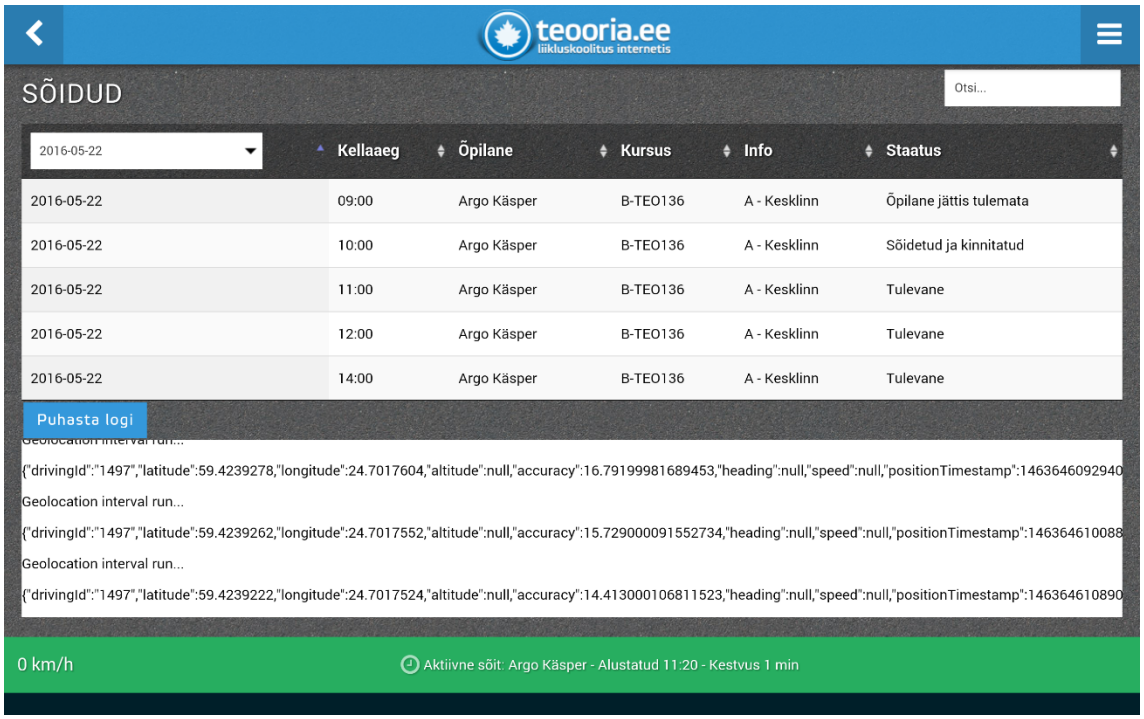
```
        translate: new Translate(),
        initialize: function() {
            this.bindEvents();
        },
        bindEvents: function() {
            document.addEventListener('deviceready',      this.onDeviceReady,
false);
        },
        onDeviceReady: function() {
            document.addEventListener('backbutton', this.backPressed, false);
            app.receivedEvent();
            if(navigator.camera) {
                app.setUpCamera();
            }
        },
       // methods continue…

    app.initialize();
});
```

Lsa 7 – Teooria mobiilirakenduse *router.js* faili näide

```
define(function (require) {
    "use strict";
    var $             = require('jquery'),
        Backbone      = require('backbone'),
        LoginView     = require('app/views/Login'),
        DashboardView = require('app/views/Dashboard'),
        ClientView    = require('app/views/Client'),
        DrivingView   = require('app/views/Driving'),
        CalendarView  = require('app/views/Calendar'),
        CameraView    = require('app/views/Camera'),
        SettingsView  = require('app/views/Settings'),
        Header        = require('app/views/Header'),
        Footer        = require('app/views/Footer'),
        DataTables    = require('jquery.dataTables'),
        JQueryUI      = require('jquery-ui'),
        JQueryUITouch = require('jquery-ui-touch-punch.min'),
        Moment        = require('moment'),
        FullCalendar  = require('fullcalendar.min'),
        Translate     = require('trans/translate');



    return Backbone.Router.extend({
        translate: new Translate(),
        initialize: function(options) {
            this.layout = options.layout;
          },

        routes: {
            ""            : "login",
            "login"       : "login",
            "home"        : "dashboard",
            "clients"     : "clientList",
            "drivings"    : "drivingList",
```

```
        "calendar"    : "calendar",
        "camera"      : "camera",
        "settings"    : "settings"
    },
    _init: function() {
        // function implementation
    },
    login: function () {
        window.sessionStorage.clear();
        this._init();
        this.layout.setHeader(new Header());
        this.layout.setContent(new LoginView());
    },
    dashboard: function () {
        this._init();
        this.layout.setHeader(new Header());
        this.layout.setContent(new DashboardView());
        this.layout.setFooter(new Footer());
    },
// methods continue…
```

Lisa 8 – GPS logi kogumine Teooria mobiilirakendusega



Lisa 9 – GPS logi kogumine Android testrakendusega

Lisa 10 – Android testrakenduse logid

[GPS] [2016-05-18T17:47:29.267+03:00] GPS tracking started! ***

[GPS] [WITH TIMESTAMPS] [lat:59.4236805, lon:24.7014604, accuracy=31.5, requestedTime: 1463582849267, receivedTime: 1463582849297, receivedTime-requestedTime=30]

--

[GPS] [WITH TIMESTAMPS] [lat:59.4231311, lon:24.6985522, accuracy=43.5, requestedTime: 1463583035868, receivedTime: 1463583035921, receivedTime-requestedTime=53]

--

[GPS] [WITH TIMESTAMPS] [lat:59.422366, lon:24.6981828, accuracy=40.5, requestedTime: 1463583047149, receivedTime: 1463583047173, receivedTime-requestedTime=24]

--

[GPS] [WITH TIMESTAMPS] [lat:59.4226458, lon:24.6981413, accuracy=45.0, requestedTime: 1463583106788, receivedTime: 1463583106808, receivedTime-requestedTime=20]

[GPS] [WITH TIMESTAMPS] [lat:59.4220049, lon:24.6979656, accuracy=34.5, requestedTime: 1463583108043, receivedTime: 1463583108080, receivedTime-requestedTime=37]

--

[GPS] [WITH TIMESTAMPS] [lat:59.4203567, lon:24.6963339, accuracy=45.0, requestedTime: 1463583169874, receivedTime: 1463583170026, receivedTime-requestedTime=152]

--

[GPS] [WITH TIMESTAMPS] [lat:59.4090964, lon:24.6856564, accuracy=346.5, requestedTime: 1463583290135, receivedTime: 1463583290167, receivedTime-requestedTime=32]

[GPS] [WITH TIMESTAMPS] [lat:59.4078711, lon:24.6885696, accuracy=39.0, requestedTime: 1463583327069, receivedTime: 1463583327081, receivedTime-requestedTime=12]

[GPS] [WITH TIMESTAMPS] [lat:59.4078139, lon:24.6887933, accuracy=39.0, requestedTime: 1463583345443, receivedTime: 1463583345472, receivedTime-requestedTime=29]

[GPS] [WITH TIMESTAMPS] [lat:59.4077848, lon:24.6888508, accuracy=37.5, requestedTime: 1463583387341, receivedTime: 1463583387384, receivedTime-requestedTime=43]

--

[GPS] [2016-05-18T17:56:29.512+03:00] GPS tracking stopped! ***

[CAMERA] [start=1463646282443, end=1463646282455, end-start=12]

[CAMERA] [start=1463646286020, end=1463646286031, end-start=11]

[CAMERA] [start=1463646289531, end=1463646289541, end-start=10]

[CAMERA] [start=1463646292993, end=1463646293002, end-start=9]

[CAMERA] [start=1463646296434, end=1463646296443, end-start=9]

[CAMERA] [start=1463646299699, end=1463646299709, end-start=10]

[CAMERA] [start=1463646303684, end=1463646303693, end-start=9]

[CAMERA] [start=1463646307228, end=1463646307240, end-start=12]

[CAMERA] [start=1463646326776, end=1463646326786, end-start=10]

[CAMERA] [start=1463646331103, end=1463646331113, end-start=10]

[NOTIFICATION] [CONFIRM] [start=1463646338829, end=1463646338843, end-start=14]

[NOTIFICATION] [CONFIRM] [start=1463646341010, end=1463646341010, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646342996, end=1463646342996, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646344759, end=1463646344759, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646347573, end=1463646347573, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646353942, end=1463646353943, end-start=1]

[NOTIFICATION] [CONFIRM] [start=1463646355865, end=1463646355865, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646357846, end=1463646357847, end-start=1]

[NOTIFICATION] [CONFIRM] [start=1463646387199, end=1463646387199, end-start=0]

[NOTIFICATION] [CONFIRM] [start=1463646392456, end=1463646392456, end-start=0]

[NOTIFICATION] [ALERT] [start=1463646410980, end=1463646410994, end-start=14]

[NOTIFICATION] [ALERT] [start=1463646413170, end=1463646413187, end-start=17]

[NOTIFICATION] [ALERT] [start=1463646414964, end=1463646414978, end-start=14]

[NOTIFICATION] [ALERT] [start=1463646416829, end=1463646416845, end-start=16]

[NOTIFICATION] [ALERT] [start=1463646418430, end=1463646418444, end-start=14]

[NOTIFICATION] [ALERT] [start=1463646447563, end=1463646447580, end-start=17]

[NOTIFICATION] [ALERT] [start=1463646449592, end=1463646449607, end-start=15]

[NOTIFICATION] [ALERT] [start=1463646451449, end=1463646451466, end-start=17]

[NOTIFICATION] [ALERT] [start=1463646453212, end=1463646453227, end-start=15]

[NOTIFICATION] [ALERT] [start=1463646454963, end=1463646454978, end-start=15]

Lisa 11 – Teooria mobiilirakenduse logid

GPS logi:

[{"drivingId":"1340","latitude":"59.4237366","longitude":"24.7014365","altitude":"","accuracy":"33","heading":"","speed":"","positionTimestamp":"1463582954089","timestamp":"1463582954181","requestSent":"1463582953666"},
{"drivingId":"1340","latitude":"59.4236785","longitude":"24.7014938","altitude":"","accuracy":"40.5","heading":"","speed":"","positionTimestamp":"1463582962769","timestamp":"1463582962858","requestSent":"1463582961667"},
{"drivingId":"1340","latitude":"59.4234205","longitude":"24.7012695","altitude":"15.546319942843358","accuracy":"8","heading":"","speed":"0","positionTimestamp":"1463582971633","timestamp":"1463582971673","requestSent":"1463582969668"},
// more data here…
{"drivingId":"1340","latitude":"59.4216788","longitude":"24.6974874","altitude":"24.024789915129055","accuracy":"6","heading":"200","speed":"0","positionTimestamp":"1463583131880","timestamp":"1463583131935","requestSent":"1463583129780"},
{"drivingId":"1340","latitude":"59.4216788","longitude":"24.6974874","altitude":"24.024789915129055","accuracy":"6","heading":"200","speed":"0","positionTimestamp":"1463583131908","timestamp":"1463583131953","requestSent":"1463583129780"}]

[CAMERA] startTime=1463645396135, endTime=1463645396188, endTime-startTime=53
[CAMERA] startTime=1463645405259, endTime=1463645405287, endTime-startTime=28
[CAMERA] startTime=1463645411173, endTime=1463645411193, endTime-startTime=20
[CAMERA] startTime=1463645422804, endTime=1463645422826, endTime-startTime=22
[CAMERA] startTime=1463645429018, endTime=1463645429039, endTime-startTime=21
[CAMERA] startTime=1463645433855, endTime=1463645433891, endTime-startTime=36
[CAMERA] startTime=1463645440221, endTime=1463645440242, endTime-startTime=21
[CAMERA] startTime=1463645448900, endTime=1463645448921, endTime-startTime=21
[CAMERA] startTime=1463645455293, endTime=1463645455317, endTime-startTime=24
[CAMERA] startTime=1463645463060, endTime=1463645463079, endTime-startTime=19

[CONFIRM] startTime=1463645556769, endTime=1463645556774, endTime-startTime=5
[CONFIRM] startTime=1463645559429, endTime=1463645559434, endTime-startTime=5

[CONFIRM] startTime=1463645561180, endTime=1463645561186, endTime-startTime=6

[CONFIRM] startTime=1463645562609, endTime=1463645562612, endTime-startTime=3

[CONFIRM] startTime=1463645564081, endTime=1463645564084, endTime-startTime=3

[CONFIRM] startTime=1463645566898, endTime=1463645566902, endTime-startTime=4

[CONFIRM] startTime=1463645573525, endTime=1463645573528, endTime-startTime=3

[CONFIRM] startTime=1463645576343, endTime=1463645576349, endTime-startTime=6

[CONFIRM] startTime=1463645578517, endTime=1463645578520, endTime-startTime=3

[CONFIRM] startTime=1463645580740, endTime=1463645580744, endTime-startTime=4

[ALERT] startTime=1463645865538, endTime=1463645865543, endTime-startTime=5

[ALERT] startTime=1463645867511, endTime=1463645867514, endTime-startTime=3

[ALERT] startTime=1463645869307, endTime=1463645869314, endTime-startTime=7

[ALERT] startTime=1463645871176, endTime=1463645871178, endTime-startTime=2

[ALERT] startTime=1463645872964, endTime=1463645872966, endTime-startTime=2

[ALERT] startTime=1463645874722, endTime=1463645874725, endTime-startTime=3

[ALERT] startTime=1463645876462, endTime=1463645876465, endTime-startTime=3

[ALERT] startTime=1463645878041, endTime=1463645878044, endTime-startTime=3

[ALERT] startTime=1463645879662, endTime=1463645879665, endTime-startTime=3

[ALERT] startTime=1463645885802, endTime=1463645885804, endTime-startTime=2