TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia Teaduskond
Informaatikainstituut
Informaatika õppetool

Firmade koosolekuruumide kasutuse optimeerimise tarkvara

bakalaureusetöö

Üliõpilane: Marko Kaar
Üliõpilaskood: 123443IAPB
Juhendaja: Ants Torim

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.

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

(*kuupäev*) (*allkiri*)

Annotatsioon

Uurimus tutvustab lean startup ning agiilse arenduse metoodikaid kasutades loodud veebirakendust, mis suudab andmeid analüüsida ning neist järeldusi tekitada. Selleks, et lugeja aru saaks mis on lean startup ja agiilse arenduse printsiibid, ning kuidas neid lõpptoote arendamisel rakendati on eelnevalt töökäigule kirjeldatud töö teoreetiline osa agiilsest arendamisest ning lean startup metoodikast.

Töö eesmärk on luua taustsüsteem mis suudab lugeda logifailides olevaid andmeid ning nende põhjal ettemääratud reegleid kasutades tekitada järeldus sellest, kas saab kuidagi optimeerida kindla kliendi kinnisvaraliste ressursside kasutust. Selle süsteemi kasutajasõbralikuks kasutamiseks on vajalik luua visuaalne liides eelmainitud andmete kuvamiseks, klientide haldamiseks, ja raportite genereerimiseks.

Põhilised probleemid mis töös tekkisid oli testimiseks vajalike andmete puudumine. Selleks, et seda probleemi lahendada sai tehtud programm mis genereerib logifailid ning lisab neid Google Calendar keskkonda kus käib klientidel koosolekute haldus. Teine probleem mis tekkis oli see, et kuna loetavaid andmeid oli võrdlemisi palju, läks programmil palju aega nende analüüsimisele. Selle lahenduseks lisati kaks asja - vahemälu funktsionaalsus mis jätab analüüsitud andmed määratud ajaks meelde, ning tagastab need kui nende aegumisperiood ei ole möödunud ning paralleelsuse tugi mis laseb mitmel päringul samal ajal joosta.

Töö tulemusena valmib veebirakendus milles on võimalik määrata analüüsitavate andmete allikas, vaadata algoritmide töö tulemusi ning graafilisi raporteid. Loodetavasti on töö läbilugemisel saanud lugeja teadmisi ka agiilsest arendusest ning lean startup metoodikatest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 5 peatükki, 13 joonist, 2 tabelit.

Abstract

Software to Optimize the Usage Of Companies' Meeting Rooms

This thesis will introduce Lean Startup and Agile development methodologies using the web application that will be created as a result of the thesis. The web application will be able to parse data from log files and draw conclusions based on the aforementioned analysis. For the reader to understand what are Lean Startup and Agile development and its principles and how they were used in the development of the final product, the thesis contains a theoretical section about Agile development principles and Lean Startup methods.

The purpose of the thesis is to create a backend system that can parse data contained in log files and using predefined rules it should be able to draw a conclusion to whether it's possible to optimize a certain client's real-estate resource usage. To make the system user-friendly a frontend interface has to be created for displaying the analysis results, client administration and the generation of reports.

The main issues that arose during the thesis was that I lacked the data that I needed for testing. To solve that problem, a script was created that generates log files and adds the generated meetings to the Google Calendar environment. Another issue that arose was that since the amount of data being analyzed was quite large, it took a while for the software to go through it all. All other requests would be in a waiting state in that time. To solve this two fixes were done – concurrency support was added and a cache was added to reduce the amount of requests being done. The cache would remember the analyzed data for a preset amount of time and returns the data to the client if their expiration date hasn't been reached.

As a result of the thesis a web application will be created in which the client can set the source of the data being analyzed, they can look at the results of the data analysis and see graphical and textual reports on the data. Hopefully whilst reading the thesis, the reader will gain knowledge about Agile development and Lean Startup methodologies.

The thesis is in estonian and contains 42 pages of text, 5 chapters, 13 figures, 2 tables.

Lühendite ja mõistete sõnastik

MVP	***Minimum Viable Product*** minimaalne töötav toode
Lõimed	***Threads*** Javasse ehitatud ülesande lahendusmehhanism
Paralleelsus	***Concurrency*** Mitme lõime samaaegne jooksutamine
šabloonimootor	***Template engine*** Tehnoloogia mis lubab serveripoolset infot kuvada kasutajaliideses
Dünaamiline veebiliides	***Responsive web*** Veebileht mis muudab oma välimust vastavalt seadmele millest seda vaadatakse
REST	***Representational State Transfer*** Arhitektuuriline stiil kuidas andmetele ligipääseda
JSON	***Javascript Object Notation*** Javascripti objektide vorming
JDBC	***Java Database Connectivity*** Javale kirjutatud andmebaaside ühendusi haldav liides
API	***Application program interface*** Rakenduse liides, mis tagab ligipääsu kindla toote funktsionaalsusele

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Tänapäeval kulutavad IT firmad väga palju raha oma kontorite peale. Mida suurem on töötajate arv seda suuremat kontorit on firmal vaja. Enamjaolt on firmadel ka rohkem kui üks koosolekutuba, jällegi mida suurem on firma, seda rohkem on vaja koosolekutubasid, et kõik vajalikud koosolekud peetud saaks. Kuidas ning kui efektiivselt neid tube kasutatakse, selle kohta üldiselt statistika puutub.

Käesolev bakalaureusetöö üritabki seda probleemi lahendada, pakkudes graafilist statistikat toas oleva tehnika kasutuse ning tubade täituvuse kohta.

1.1 Taust ja probleem

Firmade piiratuim ressurss on kinnisvara. Seda ei saa väga lihtsalt juurde osta ning selle jaotuse üle väga tihti suuremat mõttetööd ei tehta. Koosolekuruumid spetsiifiliselt erinevad oma olemusest – ühed on mõeldud suuremate gruppide ühisteks nõupidamisteks, teised videokonverentsideks, kolmandad koolituste korraldamiseks. Üldiselt broneerivad töötajad enda vajadustele vastava koosolekuruumi kuid kindlasti leidub ka kordi kus koosolekutuba võetakse inimese enda mugavusele lähtudes, ehk siis selline tuba mis on töötajale kõige lähemal ning milles olevat tehnikat tal reaalselt vaja ei lähe. Sellise informatsiooni kohta puudub firmadel tihti ülevaade, kuna puuduvad selleks vajalikud tööriistad.

Toode mis bakalaureusetöö raames valmib on suunatud firmadele kellel on mitu koosolekuruumi ning huvi teada saada kuidas töötajad neid kasutavad.

Bakalaureusetöö valmib vahemikus 2016. aasta kevadel autori õpingute ajal Tallinna Tehnikaülikoolis ning ametiaja jooksul firmas Nortal AS.

1.2 Ülesande püstitus

Bakalaureusetöö tulemusena luuakse tarkvara kasutades lean startup metoodikaid ning agiilse arenduse põhimõtteid, mis suudab etteantud logifailidest välja parsida andmed koosolekutubade kasutusest, nende andmete analüüsi põhjal luua järeldusi koosolekutoa kasutusest ning sellest kas seda saab kuidagi optimeerida, ning nende järelduste põhjal peab suutma programm tekitada visuaalsed graafikud, mis kuvatakse veebiliideses. Bakalaureusetöö on täies mahus koostatud ning programmeeritud ainuüksi autori poolt.

1.3 Metoodika

Bakalaureusetöö kirjutamisel tutvutakse kirjandusega, mis toetab töö teoreetilise osa eesmärke ning toote analüütilisi lahendusi.

Tagarakendus luuakse Java keeles, kasutades Spring MVC raamistikku. Veebiliidese loomiseks kasutatakse CSS raamistikku Bootstrap ning visuaalsete andmete loomiseks JavaScriptil põhinevaid teeke.

Selleks, et veenduda äriloogika korrektses töös lisatakse rakendusele automaattestid. See on suur osa agiilse arenduse põhimõttest ning tänu automaattestidele võime kindlad olla, et andmed töödeldakse korrektselt ning rakendus tagastab õiged andmed.

1.4 Ülevaade tööst

Bakalaureusetöö alustuseks selgitatakse lugejale mis asi on agiilne arendus ning lean startup. Selle teadmisega saab lugeja ülejäänud tööprotsessist paremini aru ning suudab jälgida miks ühte või teist tehnikat toote arendamisel kasutati.

Töö teises pooles kirjeldatakse toote valmistamise protsessi ning põhjendatakse miks valiti kasutatud tehnoloogiaid. Samuti kirjeldatakse toodet ennast ning demonstreeritakse kuidas kindlad lehed välja näevad ning mis formaadis andmeid veebiliidese ning tagarakendi vahel vahetatakse.

2. Teoreetiline taust

2.1 Lean Startup

Kuidas luua toodet millel on võimalik turul oma koht leida? Kindlasti seisneb sellises protsessis toote- ning turuanalüüs, toote põhjalik disainimine, kuid isegi kui toode on tehniliselt hästi valmistatud ei garanteeri see toote edukust turul. „Liiga paljud idufirmad saavad alguse mõttega, et nad loovad toodet mida inimesed tahavad kasutada. Nad veedavad kuid, vahest aastaid, et seda toodet täiuslikuks muuta, ilma kasvõi väga toorest toodet potentsiaalsele kliendile näitamata. Kui nad ei suuda toode vastu huvi kasvatada on see üldiselt sellepärast, et nad ei olnud kordagi tulevaste klientidega rääkinud ega vaadanud kas nende toode on reaalselt huvitav. Kui kliendid lõpuks suhtlevad, tänu nende ükskõiksusele toote vastu idufirma ebaõnnestub.“ [1]

Lean startup on metoodika mis loodi 2008 aastal Eric Riesi poolt. Ries on mitme ameerika idufirma konsultatiivnõukogus, ning Harvardi ärikooli ettevõtluse juhendaja. [2] Lean startup on arendusmetoodika, mis sarnaselt teistele arendusmetoodikatele on tsükliline. Kuidas ta teistest metoodikatest erineb on see, et ta on pigem suunatud idufirmadele ning on palju kompaktsem kui tavaline arendusprotsess. Metoodikal on kolm põhi sammu: idee või visiooni defineerimine, selle programmeerimine, ning tulemuste mõõtmine ja neist õppimine. Protsessi alguspunktiks on idee ning visioon. Uuritakse kas sellisele ideele on üldse turgu, ning kui tundub, et idee on arendust väärt hakatakse seda arendama. Arenduse jooksul tehakse valmis minimaalne toode mis täidab eesmärki ning hakatakse otsima potentsiaalseid kliente kes sellist toodet kasutaks. Arendusprotsessi järel kogutakse andmeid sellest kui mitu klienti oleks arendatavat toodet reaalselt kasutanud ning kui idee jaoks turgu ei ole ning kliente ei leita, loetakse toode ebaõnnestunuks, õpitakse tulemustest ning liigutakse edasi järgmise idee juurde ning tsükkel jätkub.

Lean startup metoodika koosneb neljast põhiprintsiibist:

1) Eemaldage ebakindlus

Enamus toodete arendamisel puudub kindlustunne, et see toode on edukas. Tihti analüüsitakse erinevaid praktikaid ning töövahendeid üle ning lõpuks läheb rohkem aega toote analüüsile kui reaalsele arendamisele. Paljud idufirmad on mõtteviisiga, et hakkame lihtsalt kusagilt

pihta ja vaatame kuhu tee meid viib. See pole alati ka kõige õigem lähenemine kuna lõpuks võib lahendus olla vägagi kulukas. Tähtis on toote visioon. Lean startup lähenemine pakub firmadele struktuuri, andes neile võimalused oma visiooni pidevalt testida. Kui tõlkida sõna *lean* inglise keelest võiks öelda, et eesti keelne vaste sellele oleks tõhus. Mis teeb siis toote loomise tõhusaks? Lean startup ütleb, et see ei ole ainult rahaliselt säästlik arendus, ega ka mitte kiire ning odav läbikukkumine. Toote loomine muutub tõhusaks siis, kui selle arenduse ümber on defineeritud protsess ning paika pandud kindel metoodika.

2) Töötage targemalt, mitte rohkem

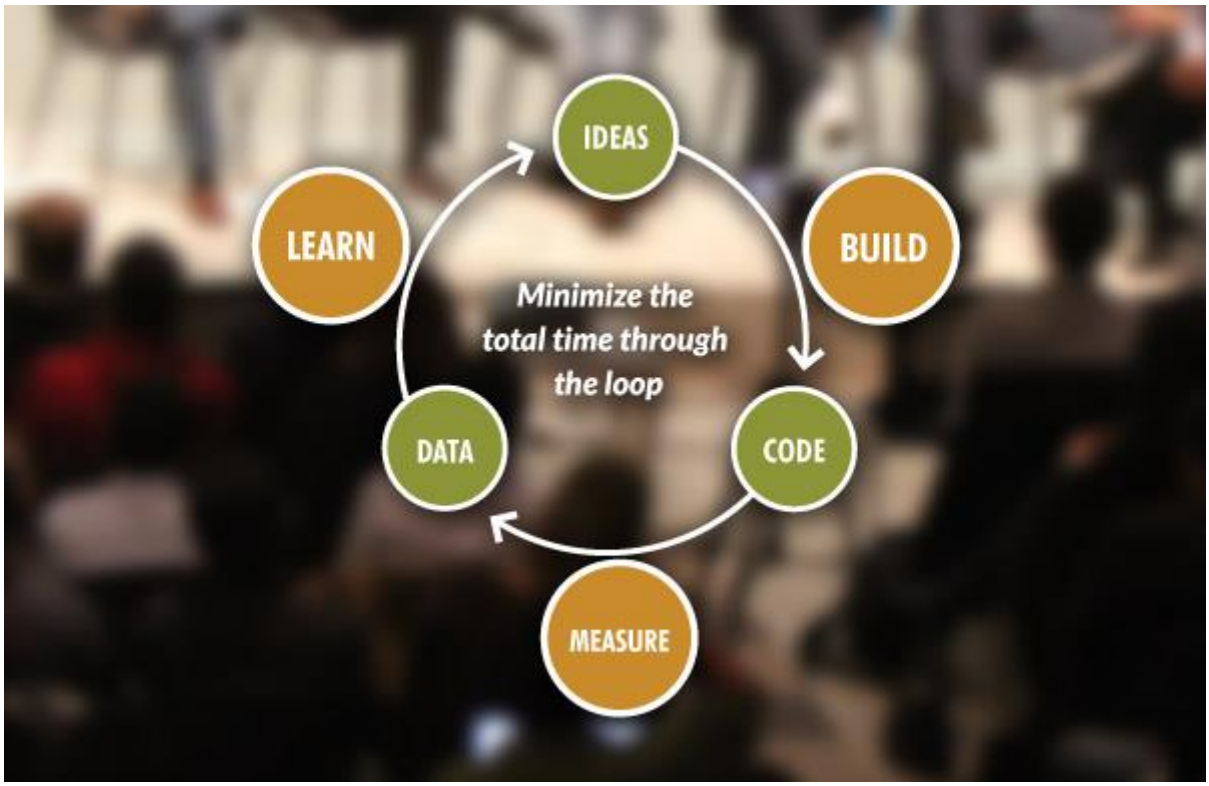
Lean startupi puhul ei ole küsimus selles, kas ning kuidas mingit toodet on võimalik toota. Pigem on küsimus selles, kas seda on mõtet üldse toota ning kas tootja suudab luua jätkusuutliku firma selle kindla toote või teenuse ümber. Kui esmane toode on edukas siis on alles mõtet seda edasi arendada ning sellele funktsionaalsust juurde ehitada.

3) Looge *MVP*

MVP ehk inglisekeelne akronüüm *minimum viable product* on minimaalne toode mis täidab visiooni eesmärki. Sellel ei pruugi olla kõige täiuslikum välimus ning funktsionaalsuses võivad olla puudujäägid, peaasi et põhifunktsionaalsus töötaks. Tähtsam on ka see, et oleks ette näidata töötav toode kui see, et toode oleks tehniliselt täiuslik. Toote esimest versiooni, ehk MVP-d näidatakse potentsiaalsetele esmastele kliendile ning kui kliendid on nõus sellist toodet edaspidi ka kasutama siis lisatakse funktsionaalsust juurde. Mõõtmise sammul kogutakse MVP põhjal tagasisidet klientidelt eduka rakenduse loomiseks.

4) Valideeritud õppimine

Kui tehastes mõõdetakse edu kõrgkvaliteetsete toodete tootmise pealt siis Lean startupi edu mõõdetakse valideeritud õppimisega. Valideeritud õppimine on see kui ettevõtja suudab näidata edu olles samas täielikus ebakindluses. Toode võib olla valmimisjärgus ning töötav kuid keegi ei tea kas see turul ka müüma hakkab. Siis kui ettevõtja suudab valideeritud õppimist rakendada siis suudab ta ka arendusprotsessi kahandada märgatavalt. Kui ettevõtja keskendub sellele, et ehitada *õiget* toodet – sellist mida kliendid tahavad ja mille eest nad on nõus maksma – ei pea ettevõtja kulutama kuid oodates andmeid sellest kas toode on edukas või mitte. Selle asemel saab ettevõtja muuta oma plaane inkrementaalselt, samm haaval. [1] See põhimõte on sarnane agiilse arenduse põhimõtetele millest räägib järgmine peatükk.



Joonis 1 – Lean startup metoodika arendusprotsess [1]

2.2 Agiilne arendus

Antud peatükis kirjeldatakse, mis on agiilne arendus, räägitakse selle tähtsamatest põhimõtetest ning mainitakse kuidas agiilset arendust kasutati antud töös loodava rakenduse arendamisel.

2.2.1 Agiilse tarkvaraarenduse manifest

Agiilne tarkvaraarendus on enamjaolt kasutusel *scrum* tiimides. Kuna praegust bakalaureusetööd arendab ainult autor ise, siis selletõttu ei ole kõik agiilsuse põhimõtted käesoleva lõputöö jaoks asjakohased. Agiilse tarkvaraarenduse manifestis on nimetatud kaksteist tähtsamat põhimõtet millest osasid üritab autor ka bakalaureusetöö arendamisel rakendada. Need põhimõtted ning lühikirjeldus kuidas neid projektis kasutati on nimetatud järgnevalt tabelis 1.

Põhimõte	Kasutus projektis
Kõige olulisem on tagada kliendi rahulolu, tarnides talle vajalikku tarkvara võimalikult kiiresti ja tihti.	Bakalaureusetöö üritatakse valmistada ning tarnida võimalikult kiirelt, modulaarselt ning kvaliteetselt.
Mõistame muutuvaid olusid, isegi kui need ilmnevad arenduse lõppjärgus. Agiilsed meetodid pööravad sellised muutused meie kliendi konkurentsieeliseks.	Juhul kui on märgata, et mingi komponent ei sobi lõpptulemusega, saab selle kiirelt ära vahetada.
Tarnime tarkvara nii tihti kui võimalik, soovitavalt iga paari nädala kuni paari kuu tagant.	Seda põhimõtet töö valmimisel ei järgita kuna töö tehakse valmis nii kiirelt kui võimalik. Selle punkti osas järgitakse rohkem *kanban* arendusmetoodikat.
Valdkonna spetsialistid ja tarkvaraarendajad peavad töötama igapäevaselt koos kogu projekti vältel.	See põhimõte ei ole asjakohane kuna arendajaid on ainult üks.
Projekti edukuse aluseks on motiveeritud inimesed. Loo neile meeldiv ja toetav töökeskkond ning nad saavad iseseisvalt tööga hakkama.	See põhimõte on suunatud suuremale meeskonnale.
Kõige tõhusam ja tulemuslikum viis info jagamiseks arendusmeeskonnas on näost näkku vestlus.	See põhimõte ei ole asjakohane kuna on ainult üks arendaja.
Edu peamiseks mõõdupuuks on töötav tarkvara.	Bakalaureusetöö edukust võib samuti mõõta töötava tarkvara mõõtepunktist.
Agiilse tarkvaraarenduse protsessid soodustavad jätkusuutlikku arendust. See	Bakalaureusetöö arendustempo on üpris kiire kuna aeg on limiteeritud. Agiilse

tähendab, et projektiga saab samas tempos jätkata määramata aja jooksul.	tarkvaraarenduse protsessid kindlasti hõlbustavad seda.
Tehnilist täiuslikkust ja head disaini pideva tähelepanu all hoides tagatakse tarkvaraarenduse kiirus ja paindlikkus.	Tehniline täiuslikkus ning hea disain tulevad tootesse tänu *clean code* põhimõtteid arvesse võttes. See tagab ka kiiruse ning paindlikuse.
Lihtsus – ebavajaliku töö tegematajätmise kunst – on väga oluline.	Kuna aeg lõputöö valmistamiseks on piiratud siis võib tulla ette, et peab leidma mõned otseteed toote tööle saamiseks.
Parimad arhitektuurilised lahendused, nõuded ja disain tekivad iseorganiseeruvates meeskondades.	See põhimõte ei ole asjakohane kuna meeskond on puudulik. Samas on arendusmeeskond vägagi iseorganiseeruv.
Meeskond otsib regulaarselt võimalusi saamaks veelgi tõhusamaks ja muudab end vastavalt vajadusele.	See põhimõte ei ole asjakohane kuna meeskond on üheliikmeline.

Tabel 1 – Agiilse tarkvaraarenduse manifest **[3]**

Mida siis agiilne arendus endast lõpuks kujutab ning mis põhimõtteid bakalaureusetöö tegemisel järgitakse? Agiilne arendus edendab muutuste tegemist projekti käigus [4]. See ei tähenda seda, et projekti faasid on täiesti kadunud – projektil on ikka analüüsifaas, disainifaas, arendusfaas ning testimisfaas, kuid kui arendusfaasis mõeldakse mingi kindla komponendi kasutus ümber, ei pea tervet toodet uuesti disainima. Projektis kohandutakse vastavalt vajadustele. Kui on näha, et mingi komponent ei saa õigeks ajaks valmis, arendatakse funktsionaalsus nii kaugele, kui vajalik või jäetakse see komponent tootest välja. See seostub ka eelmises peatükis kirjeldatud *MVP* ideele.

3. Analüüs

3.1 Eesmärgid

Selleks, et kliendile midagi näidata mille kohta tagasisidet saada peame esmalt defineerima minimaalse töötava toote eesmärgid, ehk mida peaks saama rakenduses teha, et aru saada kas toode on turule sobiv.

- Kasutaja suudab siseneda veebikeskkonda oma kasutajanime ja parooli kasutades
- Kasutaja näeb temaga seotud firmade koosolekutubasid
- Kasutaja saab lisada süsteemi tema firmaga seonduvaid koosolekutubasid
- Kasutaja näeb koosoleku tubade andmeid ja seadmeid ning saab neid muuta
- Kasutaja näeb graafikuid ning raporteid koosolekutoa kasutuse kohta.

Kui need eesmärgid on täidetud saab minna potentsiaalsete klientide juurde toodet demonstreerima ning kui toode neile huvi pakub ning nad oleksid nõus seda ostma siis tehakse edasised arendused.

3.2 Eeldused

Selleks, et töös arendatav veebirakendus töötaks on vaja kindlaid riistvaralisi seadmeid mis jäävad lõputöö ning toote enda arenduse skoobist välja. Need seadmed on järgnevad:

- Koosolekutuppa sisenemise ning väljumise sensor, ehk seade mis loendab mitu inimest koosolekul käis
- Igal koosoleku ruumis oleval seadmel peaks olema lihtne sensor mis vaatab kas seda seadet lülitati koosoleku kestvuse jooksul sisse, see annab ülevaate mis seadmeid toas reaalselt kasutati.

Nende seadmete poolt raporteeritavaid andmeid analüüsib selle projekti raames valmiv tarkvaraline toode.

3.3 Ärireeglid

- Kasutaja peab andmete nägemiseks ning lisamiseks olema ennast autentinud.
- Kasutaja peab suutma lisada koosolekuruume mida veel süsteemis ei eksisteeri.
- Kasutaja näeb vähemalt, ning ainult, viimase kolme kuu andmeid koosolekuruumide kasutuse kohta.

3.4 Reaalsed kasutusjuhud

Kasutusjuht: Kõikide koosolekutubade vaatamine

Primaarne tegutseja: Klient

Osapooled ja nende huvid:

- Klient soovib vaadata koosolekutubasid ning sellega seotud seadmeid.

Käivitav tingimus: Kliendil tekib infovajadus koosolekutubade ning nende seadmete kohta

Eeltingimus: Klient on registreerinud kasutajaks, kasutaja on seotud kindla firmaga ning on süsteemi sisse logitud.

Järeltingimus: Kliendile kuvatakse koosolekutubade ning nende seadmete kohta informatsiooni.

Stsenaarium(tüüpiline sündmuste järjestus):

1. Klient registreerib ennast kasutajaks
2. Klient määrab enda kasutajaga seonduvad firmad
3. Klient logib sisse
4. Süsteem kuvab kliendile valitud firmaga seonduvad koosoleku tubade informatsiooni

Kasutusjuht: Koosolekutoa raportite vaatamine

Primaarne tegutseja: Klient

Osapooled ja nende huvid:

- Klient soovib vaadata koosolekutoa kohta genereeritud raporteid ning graafikuid

Käivitav tingimus: Kliendil tekib infovajadus koosolekutubade kasutuse kohta

Eeltingimus: Klient on sisse loginud ning valinud koosolekutoa mille kohta infot kuvada

Järeltingimus: Kliendile kuvatakse koosolekutoa kohta genereeritud andmeid

Stsenaarium(tüüpiline sündmuste järjestus):

1. Klient logib süsteemi sisse läbi kasutajaliidese
2. Klient valib koosolekutubade nimekirjast ühe, mille kohta ta informatsiooni soovib näha
3. Süsteem kuvab kliendile valitud koosolekutoa kohta genereeritud graafikuid ning raporteid.

Kasutusjuht: Koosolekutoa lisamine ning muutmine

Primaarne tegutseja: Klient

Osapooled ja nende huvid:

- Klient soovib lisada puuduvaid koosolekutubasid või olemasolevat koosolekutoa andmeid muuta

Käivitav tingimus: Klient on valinud koosolekutoa mille kohta andmeid muuta või vajutab lisamise nuppu

Eeltingimus: Klient märkab, et süsteemist puudub kindel koosolekutuba või süsteemis oleva toa andmeid ei ole korrektsed

Järeltingimus: Kliendiga seotud koosolekutubade andmed uuendatakse või lisatakse

Stsenaarium(tüüpiline sündmuste järjestus):

1. Klient logib sisse
2. Klient näeb, et koosolekutoa informatsioon ei ole korrektne

3. Klient muudab koosolekutoa andmeid, et need vastaks tegelikkusele
4. Süsteem muudab andmebaasis andmed

Kasutusjuht: Koosolekutoa seadmete lisamine ning muutmine

Primaarne tegutseja: Klient

Osapooled ja nende huvid:

- Klient soovib lisada koosolekutoale seadmeid mis toas on kuid süsteemis ei eksisteeri

Käivitav tingimus: Klient on valinud koosolekutoa mille seadmeid soovib muuta

Eeltingimus: Klient märkab, et süsteemis puudub informatsioon koosolekutoa seadmete kohta

Järeltingimus: Valitud koosolekutoa seadmete informatsioon uuendatakse

Stsenaarium(tüüpiline sündmuste järjestus):

5. Klient logib sisse
6. Klient näeb, et koosolekutoa seadmete informatsioon ei ole korrektne
7. Klient muudab koosolekutoa seadmete andmeid, et need vastaks tegelikkusele
8. Süsteem muudab andmebaasis andmed

3.5 Tehnilised nõuded

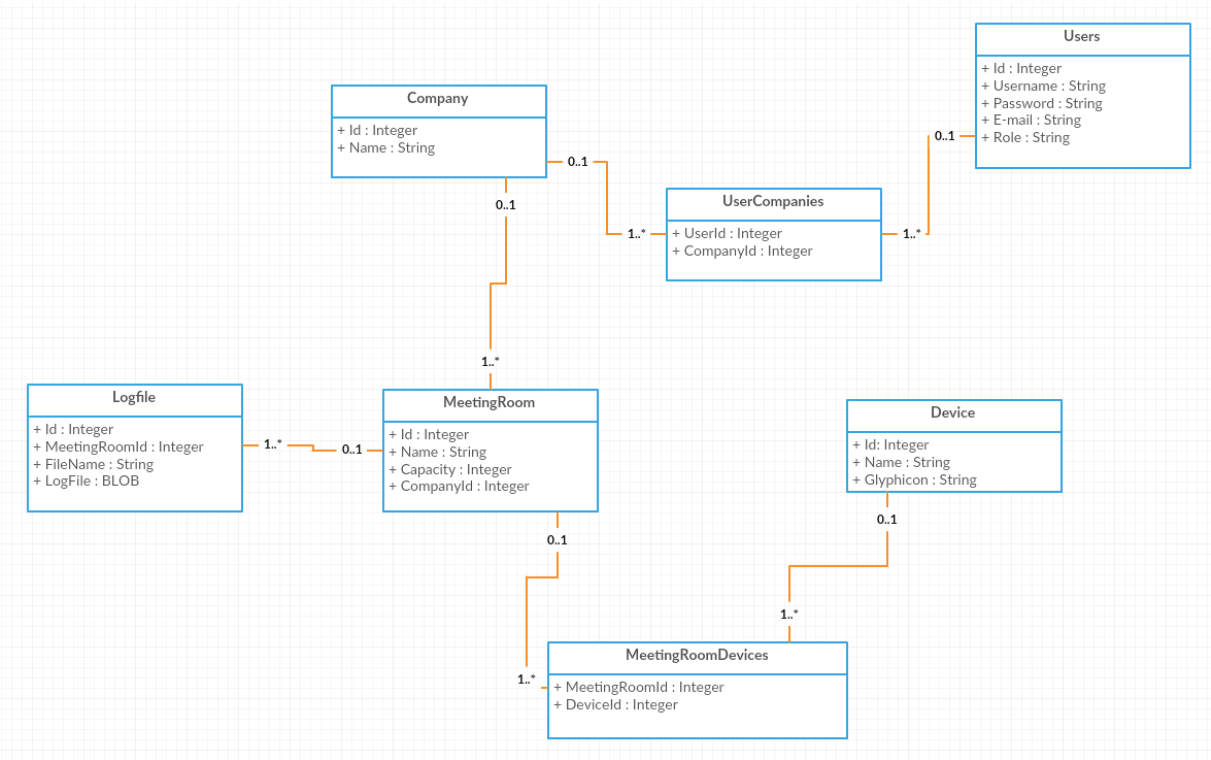
3.5.1 Veebiserver

Selleks, et rakendus kusagil jookseks on vaja veebiserverit. Veebiserveriks võeti tootel kasutusele Tomcat server. Valikus oli kas Jetty või Tomcat serverid ning kuigi Jetty serverid vajavad väidetavalt vähem hooldust siis Tomcat serveril on rohkem dokumentatsiooni ning üldiselt kasutatakse seda rohkem veebirakenduste arendamisel. Spring boot raamistikul on ka sisse ehitatud nii Jetty kui ka Tomcat serveri toetus kuid üldine arenduskogukond toetab pigem Tomcat kui Jetty rakenduse ning see oligi põhjus miks sai Tomcat server valitud.

Põhjus miks valiti just Java põhised veebiserverid on see, et toote põhieesmärk on logifaile lugeda ning nende sisu põhjal järeldusi luua. Sellist keerulist loogikat oli arendajal kõige mugavam ning mõistlikum teha just Javas.

3.5.2 Andmebaas

Kuna suurem osa lõputöös kasutatavast rakendusest tegeleb logifailide parsimisega siis väga keerulist andmebaasi projektis vaja ei ole. Põhiliselt kasutatakse andmebaasi kasutajate, firmade, koosoleku tubade ning nende kõigi omavaheliste seoste defineerimiseks (Joonis 2) siis ei ole ka vaja liialt keerulist andmebaasi süsteemi. Esimene töötav toode tuleb niikuinii ühel serveril jooksev rakendus mitte klaster seega ei ole vajadust eraldi andmebaasi serverit seadistada. Lisas 2 on defineeritud SQL laused mida andmebaasi loomiseks kasutati.



Joonis 2 – H2 Andmebaasiobjektide klassidiagramm

Logifailide hoidmiseks oli kaks valikut. Kas kirjutada nad arvuti kettale, või siis hoida neid andmebaasis. Otsustati, et struktuurse segaduse vähendamiseks hoitakse logifaile BLOB failitüübina andmebaasis. Sellisel juhul on kõik koosolekutubadega seotud andmed hajutatud Google Calendar keskkonna ning lokaalse andmebaasi vahel. Samuti tagab see andmete turvalisuse ning lihtsa ligipääsu.

Lõpuks otsustus valituks H2 andmebaas. H2 on relatsiooniline andmebaasi haldussüsteem mis on kirjutatud Javas. Projekti sai see lisatud põhjusega, et selle digitaalne jalajälg on kõige väiksem teiste süsteemidega võrreldes. Teine põhjus miks valiti H2 on see, et ta on avatud lähtekoodiga ning seda on väga lihtne implementeerida tänu selle veebiliidesele ning JDBC APIle.

3.5.3 Google Calendar

Koht kus koosolekuruume ning nende koosolekuid hallatakse on Google Calendar keskkond. Üldiselt kasutatakse koosolekutubade halduseks tihti Microsoft Exchange keskkonda kuid selle integreerimine ning sealt andmete kätte saamine bakalaureusetöö raames oleks üpris keeruline kuna sellel puudub lihtne liidestus. Selletõttu valiti test keskkonnaks Google Calendar keskkond kus on koosolekutoad defineeritud kalendritena ning koosolekud nendes kalendrites olevate sündmustena.

3.5.4 Andmete analüüsimine

Toote põhieesmärk on andmete analüüsimine, sellest räägib järgnev peatükk lähemalt.

Andmete analüüsimine käib läbi mitme protsessi. Esmalt loetakse kõik logifailid programmi mällu ning tehakse vastavad päringud Google Calendar keskkonda. Sellega viiakse kokku informatsioon koosolekutest ning sellest kui mitu inimest koosolekutel kutsunutest osales. Seejärel vaadatakse kas koosoleku ruumis olevaid seadmeid kasutati, kui ei kasutatud siis otsitakse alternatiivne koosolekutuba kuhu oleks võinud töötajad asuda. Joonisel 3 kirjeldatud kasutusdiagramm selgitab kasutatavat analüüsimiseks tehtavat eeltööd.

Logifailidest loetakse andmeid Java BufferedReader ning InputStream abil:

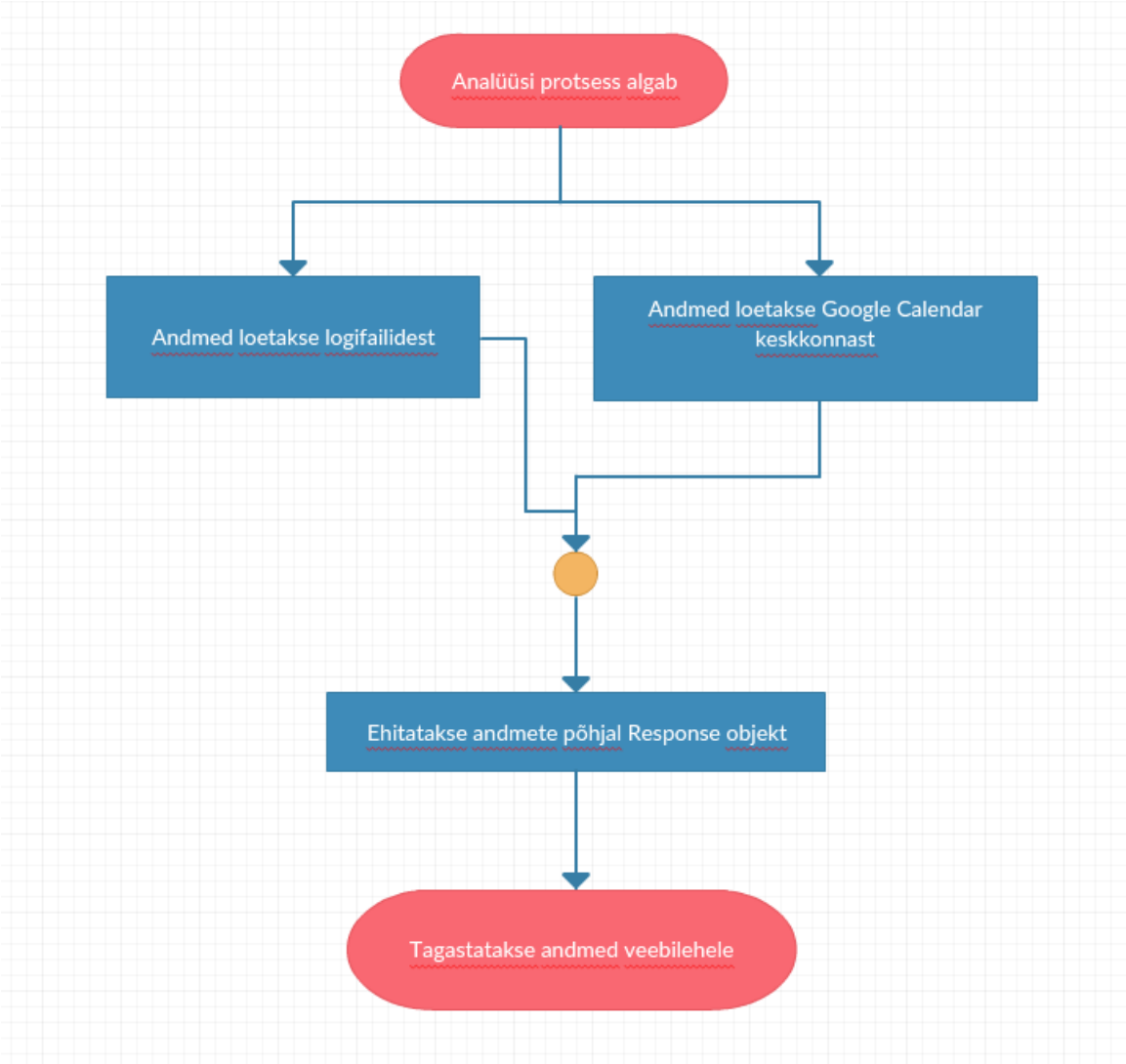
```
FileInputStream fstream = new FileInputStream(fileName);
BufferedReader br = new BufferedReader(new InputStreamReader(fstream));
String strLine;
while ((strLine = br.readLine()) != null)   {
      lines.add(strLine);
}
fstream.close();
```

Iga logifaili rida loetakse algselt mällu ning tehakse andmetest andmetöötlusobjektid. Logifaili struktuur on näidiseks toodud Lisas 1. Sellele järgnevalt tehakse Google Calendar API-sse järgnev päring:

```
Calendar service = new Calendar.Builder(httpTransport, jsonFactory, credentials)
    .setApplicationName("applicationName").build();
String pageToken = null;
do {
  Events events =
  service.events().list('primary').setPageToken(pageToken).execute();
  List<Event> items = events.getItems();
  pageToken = events.getNextPageToken();
} while (pageToken != null);
```

Tänu sellele, et projektis on kasutusel Maveni *dependency management* siis oli Google API lihtsasti integreeritav.

Pärast seda, kui andmed on programmisiseselt kättesaadavad, hakkab analüüsiprotsess pihta.



Joonis 3 - Andmete analüüsi kasutusdiagramm

Analüüsitud andmetest kuvatakse kasutajale järgnev informatsioon iga koosolekutoa kohta kuu raames, kolme kuu jooksul nagu sai ärireeglites defineeritud:

- Keskmine koosolekutele ilmumise protsent
- Keskmine koosolekutele hilinemise protsent
- Protsent ajast millal koosolekutuba on kasutuseta
- Protsent koosolekutest kus kasutati iga koosolekutoaga seotud seadet

Samuti visualiseeritakse graafikul iga päeva kohta keskmist koosolekutoa külastajate arvu. Selle arvutamiseks võetakse summa kõigist koosolekute külastajatest ning jagatakse läbi koosolekute arvuga.

Lõpuks luuakse ka järeldused eelnevalt defineeritud andmetest. Need järeldused on programmi sisse kirjutatud ning järelduste kuvamiseks arvutatakse kokku koefitsient eelnevalt toodud parameetritest mille järgi siis kuvatakse ettekirjutatud järeldus.

Veebiliidesele tagastatakse need järeldused JSON formaadis järgneva struktuuriga:

Välja nimi	**Välja tüüp**	**Andmeallikas**
Koosolekutoa nimi	String	Andmebaas
Perioodi alguskuupäev	String (date)	Logifail
Perioodi lõppkuupäev	String (date)	Logifail
Graafiku x telje tähistus	String array	Logifail
Graafiku punktid	Integer array	Logifail
Järeldused	String array	Analüüsi algoritm
Ilmumiste protsent	Integer	Logifail + Andmebaas + Google Calendar
Hilinemiste protsent	Integer	Logifail
Kasutamata aja protsent	Integer	Logifail + Google Calendar
Seadmed	String array	Andmebaas
Seadmete kasutuse protsent	Integer array	Logifail + Andmebaas

Tabel 2 - Andmete JSONi väljad ning tüübid

JSON-is näeks need andmed välja järgnevalt:

```
{
    "meetingRoomName": "Alt",
    "fromDate": "2016-05-01",
    "toDate": "2016-05-31",
    "dataLabels": [
        "02.05", "03.05", "04.05", "05.05", "06.05",
        "09.05", "10.05", "11.05", "12.05", "13.05",
        "16.05", "17.05", "18.05", "19.05", "20.05",
        "23.05", "24.05", "25.05", "26.05", "27.05"
    ],
    "dataPoints": [
        12, 11, 8, 15, 11, 10, 5, 2, 14, 20, 1, 0,
        0, 19, 3, 3, 4, 10, 11, 0
    ],
    "conclusions": [
        "Kuna selles toas ei ole ühegi koosoleku ajal kasutatud
projektorit siis ei ole seda siin toas vaja. Kui teil on suuremaid
tubasi kus puudub projektor siis võib seda seal rohkem vaja minna.",
        "Üldiselt, on see tuba kasutuseta. Tarkvara leidis, et toad
Chitchat ning Ctrl olid samadel aegadel vabad. Soovitaks sellele
toale leida teine eesmärk."
    ],
    "appearancePercent": 95,
    "latenessPercent": 0,
    "percentOfIdleTime": 50,
    "devices": [
        "projektor",
        "televiisor",
        "videokonverents"
    ],
    "deviceUsage": [
        0,
        90,
        80
    ]
}
```


3.5.5 Vahemälu

Andmete analüüsimine võtab paratamatult aega. Selleks, et süsteem ei oleks üle koormatud on mõistlik analüüsitud tulemused vahemällu salvestada ning neid säilitada üks ööpäev. Siis, kui klient andmete kuvamise lehelt ära navigeerib ning teist korda lehele naaseb, ei pea süsteem uuesti andmeid analüüsima vaid saab kiirelt juba olemasolevad andmed edastada. Lõputöös kasutati Springi raamistiku poolt pakutavat Cache objekti.

3.5.6 Paralleelne teostus

Kuna potentsiaalselt võib olla samaaegseid kasutajaid rohkem kui üks, ei saa üks klient oodata seni kuni teisele kliendile infot kuvatakse. Õnneks on Javas väga tugev paralleelsuse tugi. Lõimeid saab panna samaaegselt lahendama ülesannet ning tänu sellele saab programm mitmele kliendile samaaegselt informatsiooni koguda ning tarnida. Õnneks iga kord kui uus päring sisse tuleb ei pea me andmeid uuesti analüüsima tänu vahemälu integratsioonile. Sellest sõltumata jookseb iga uus päring eraldi lõimel ning suudab logifailidest vajaliku info samaaegselt välja lugeda, andmeid analüüsida ning genereerida kasutajatele raporteid.

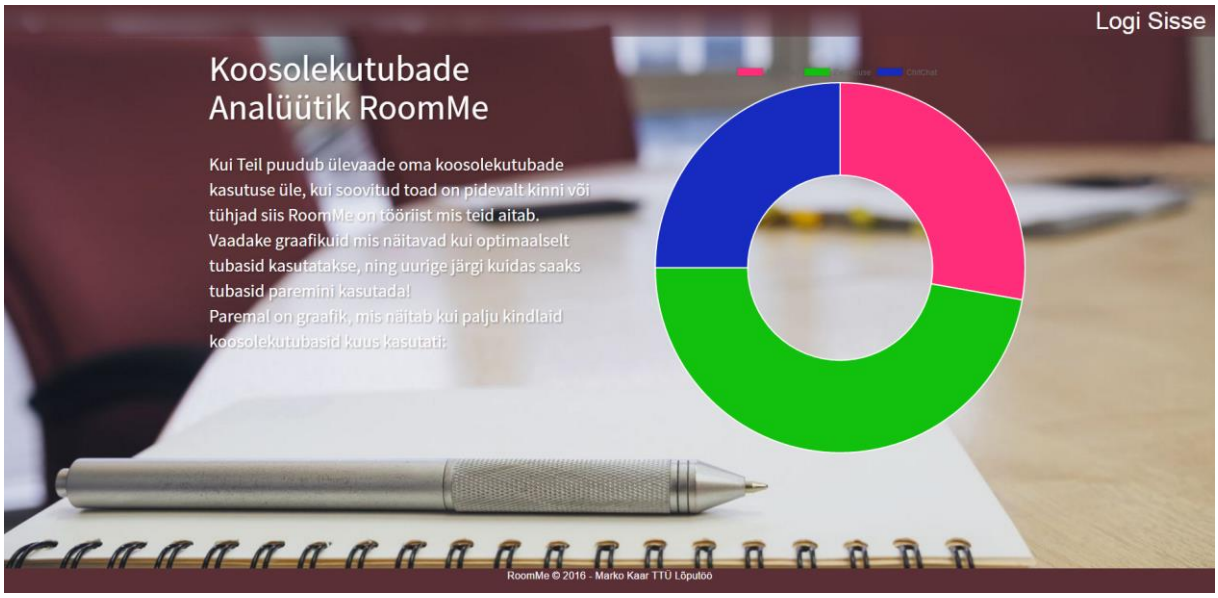
3.5.7 Kasutajaliides

Kasutajaliidese loomisel lähtuti ehituslihtsusest. Kasutati kahte põhilist tehnoloogiat – Thymeleaf mis on šabloonimootor ning Bootstrap raamistiku kasutajaliidese vormimiseks ning arenduse poolest lihtsaks dünaamilise veebiliidese toeks. Kuna me saadame võrdlemisi suures koguses andmeid läbi REST teenuste JSON-is siis kindlasti kasutame ka jQuery raamistiku mis lubab asünkroonselt andmetele ligi pääseda ning muudab veebilehe laadimise sujuvamaks, andes võimaluse laadida ressursse juba enne seda kui kasutaja on teinud valiku neid kuvada. Samuti on jQuery raamistik vajalik Bootstrap-i kindla funktsionaalsuse tööks. Lisaks sellele kasutati avatud lähtekoodiga javascripti teeki Chart.js andmete graafiliseks kuvamiseks.

3.5.8 Kasutajaliidese disain

3.5.8.1 Esileht

Kasutajaliidesel on kaks osa: maandumisleht mida kuvatakse kui kasutaja on väljalogitud ning mis tutvustab toodet ning demonstreerib raportite lehel olevaid graafikuid ning süsteemi kasutaja lehed kus asub toote põhifunktsionaalsus. Toote nimeks sai pandud RoomMe.

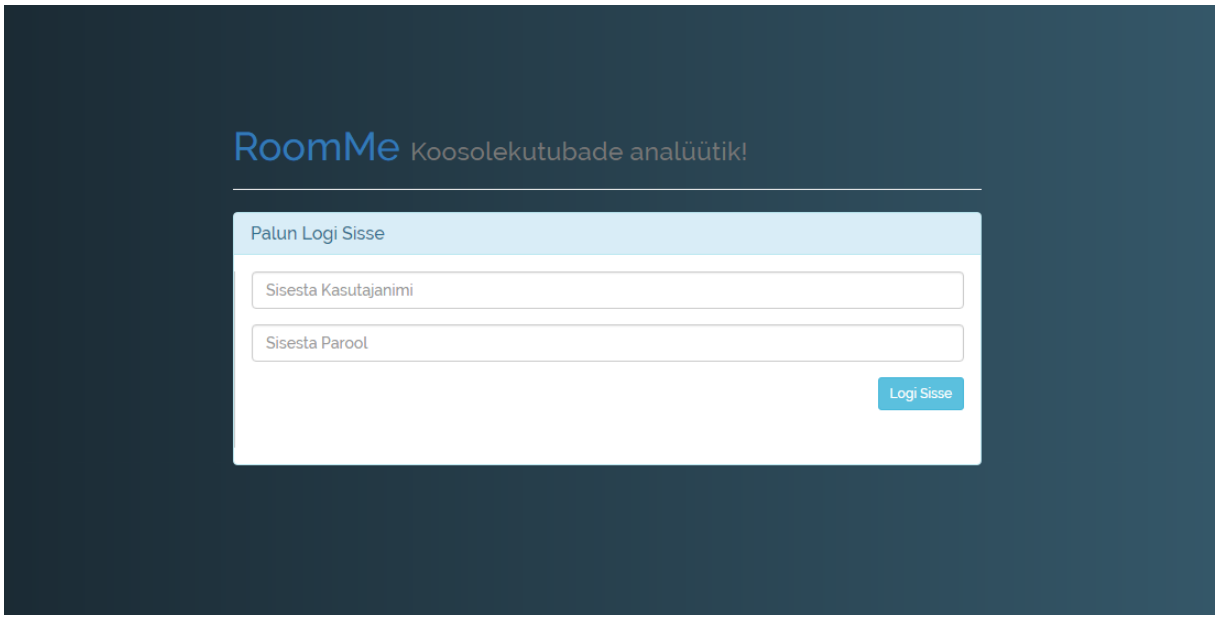


Joonis 4 - Kasutajaliidese maandumisleht

Esileht on võrdlemisi minimalistlik esimeses MVP-s, ning toob esile kasutaja jaoks tähtsamad elemendid. Nendeks on sõõriku kujuline graafiku näidis mida kasutatakse osade analüüsitud andmete kuvamisel ning toote kirjalik tutvustus.

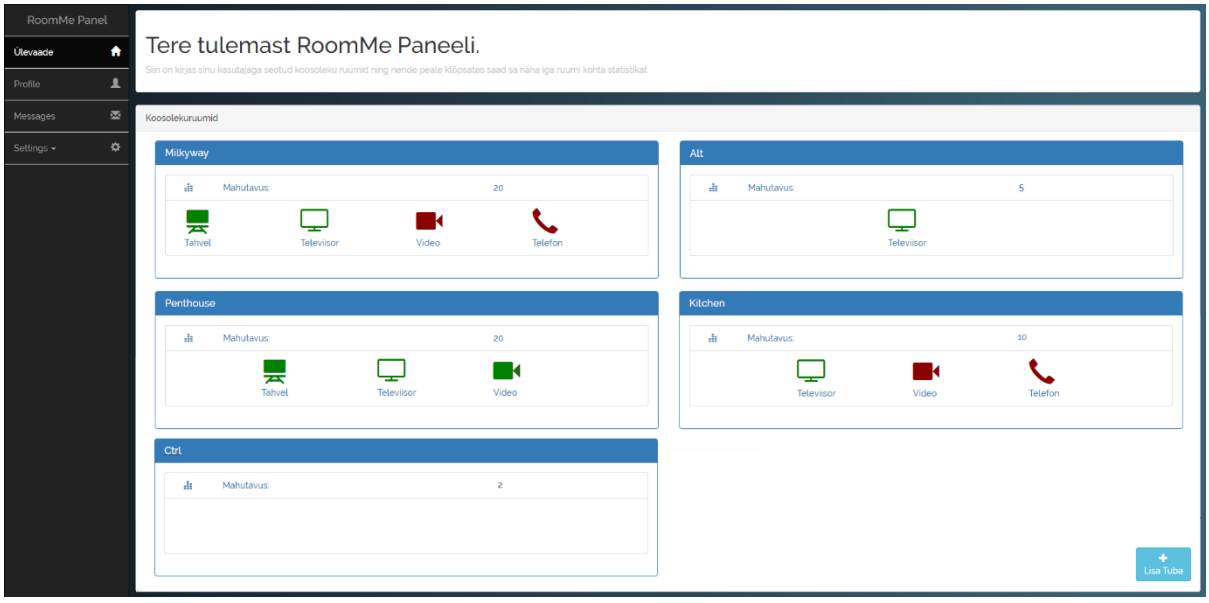
3.5.8.2 Kasutaja paneel

Selleks, et klient saaks süsteemi kasutada, peab ta eelnevalt sisse logima. Sellel eesmärgil loodi sisse logimise leht. Sisse logimine on vägagi minimaalne ning kasutab Springi raamistiku sisse ehitatud Spring Security komponenti. Selle implementatsioonis võrreldakse andmebaasis olevaid kasutajaid ning nende krüpteeritud paroole vormis sisestatud andmetega. Kui andmed on vastavuses siis suunatakse klient edasi kasutajate lehele.



Joonis 5 - Kasutaja sisse logimine

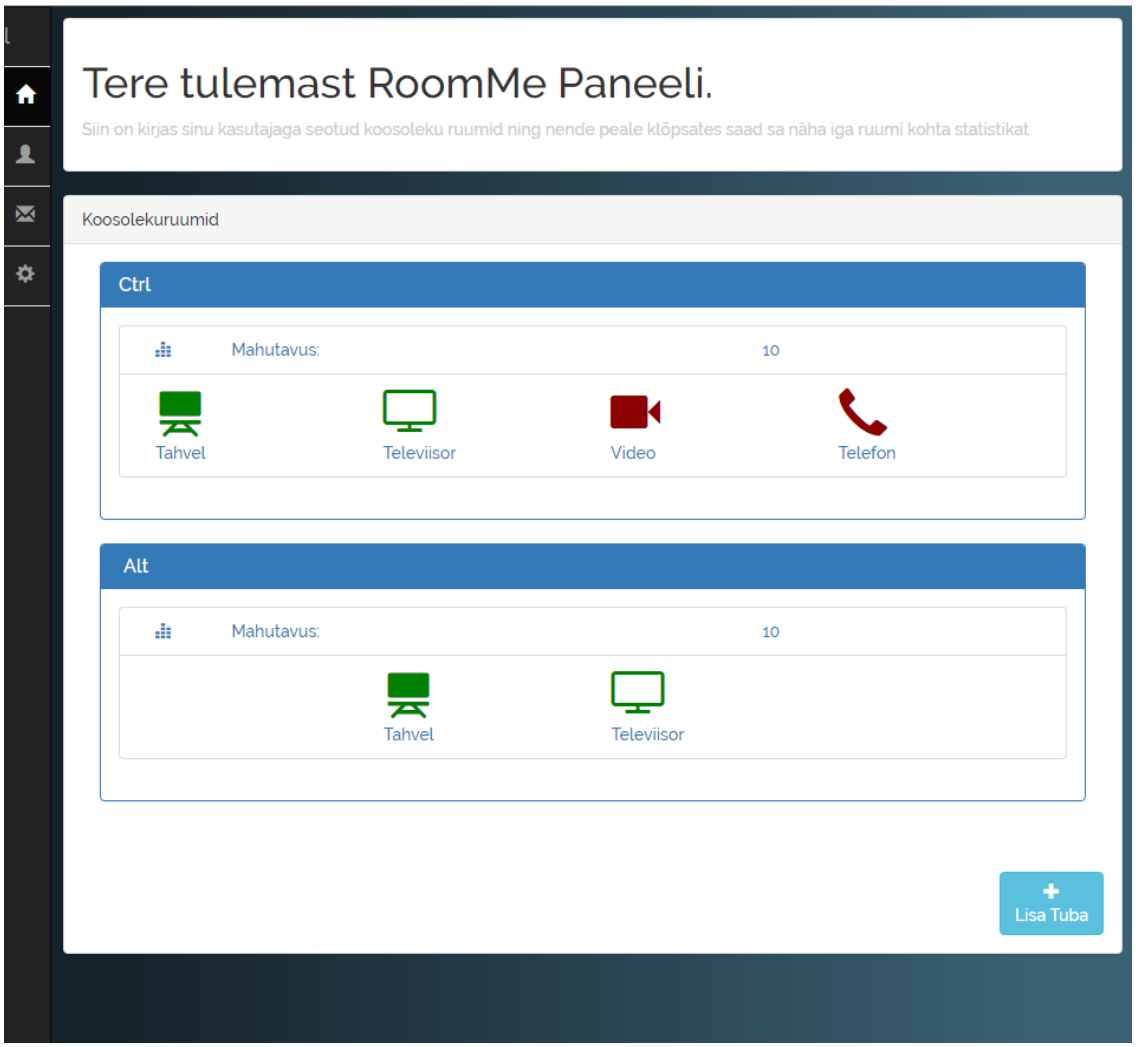
Kasutaja paneeli esilehel on näha kasutaja firmaga seotud koosolekuruumid. Iga ruumi all on märgitud selle ruumiga seotud seaded. Kusjuures kasutaja saab seadmeid ise lisada, kui sellist seadet juba süsteemis ei eksisteeri. Koosolekutoa kasti peale vajutades navigeerib kasutaja koosolekuruumi andmete lehele, kus ta näeb ruumi käesoleva kuu kasutuse andmeid.



Joonis 6 - Kasutaja paneel

3.5.8.3 Dünaamilised lehed

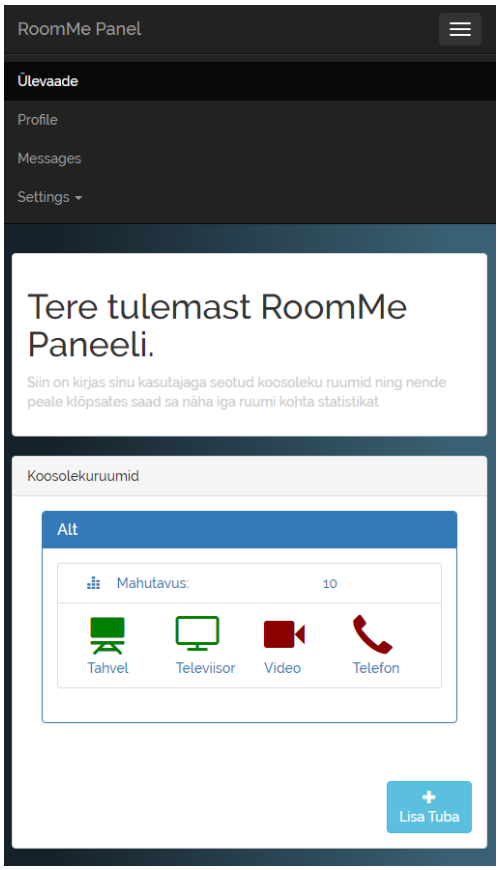
Joonisel 6 on kujutatud leht siis kui seda vaadatakse veebilehitsejast täisekraanil. Siis kui lehte avataks mõnest tahvelarvutist näeks leht välja sarnane. Navigatsiooniriba on vajunud kinni vasakule ning koosolekutubade kastid on ühes veerus üksteise all (Joonis 7).



Joonis 7 - Tahvelarvuti vaade

Joonisel 8 on kujutatud mobiilivaade, sellelt on näha, et navigatsiooniriba on liikunud lehe algusesse ning see on nähtavaks lülitatav kasutades niinimetatud hamburger-stiilis nuppu. Lehe sisu on samasugune tahvelarvuti vaatega, kus kõik koosolekutubade kastid on üksteise alla paigutatud. Selleks, et sellist vaadete dünaamilisust luua pidi vaid defineerima *bootstrap*-i klassid HTML koodis [5].

Tegemist on veergudega, inglise keeles column, ning sellest tulenevad ka klassi nimetused. Bootstrap jaotab oma sisu hoidjad – praegusel juhul div element – 12ks. Kui koodis on col-md-6 siis tähendab see seda, et tekitatakse veerg keskmise või suurema suurusega ekraanile (md ehk medium) mis on 6 ühikut lai, ehk pool hoidja laiust.



Joonis 8 - Mobiilivaade

4. Andmete haldus

4.1 Testandmete genereerimine

Lõputöö üks keerulisemaid osasid oli test andmete saamine. Kuna ligipääsu päris andmetele ei olnud siis otsustati, et peaks andmed ise genereerima. Logifailide tegemine iseenesest keeruline ei ole. Javas on alates Java 1.7st failide loomise ning nendesse kirjutamise eest hoolt kandvad teenused juba sisse kirjutatud. Oli vaja ainult defineerida šabloon mille järgi logikirjeid luua ning ülejäänud on tavaline teksti manipuleerimine. Logifailidesse tekiksid kaks rida iga koosoleku kohta – rida koosoleku alguse kohta (Joonis 9) ning rida koosoleku lõpu kohta (Joonis 10).

```
String entryTemplate = "$day.0$month.2016|Start $starth:$startm:00.000|End
$endh:$endm:00.000|$room|ID$identifier|$status|$people";
```

Joonis 9 - Koosoleku alguse logikirje šabloon

Joonisest 9 on näha, et muutujad on märgitud $ tähisega. Iga eeldefineeritud koosoleku ruumi kohta tehakse enamuste tööpäevade kohta juhusliku kellaajaga kirje, ruumi nimi milles tegevus toimub, staatus kas tegu on sisenemise või väljumisega, ning loenduri poolt saadud sisenetud inimeste arv. Kuna tegemist peaks olema loenduriga seotud logidega siis on igal kirjel algus aeg ning lõpp aeg millal loendur tööle hakkas ning millal loendamise lõpetas. Kui inimene siseneb pärast seda aega siis jääb ta sisenemis logist välja kuid pannakse kirja väljumise logisse. Nii saab leida hilinejate hulga.

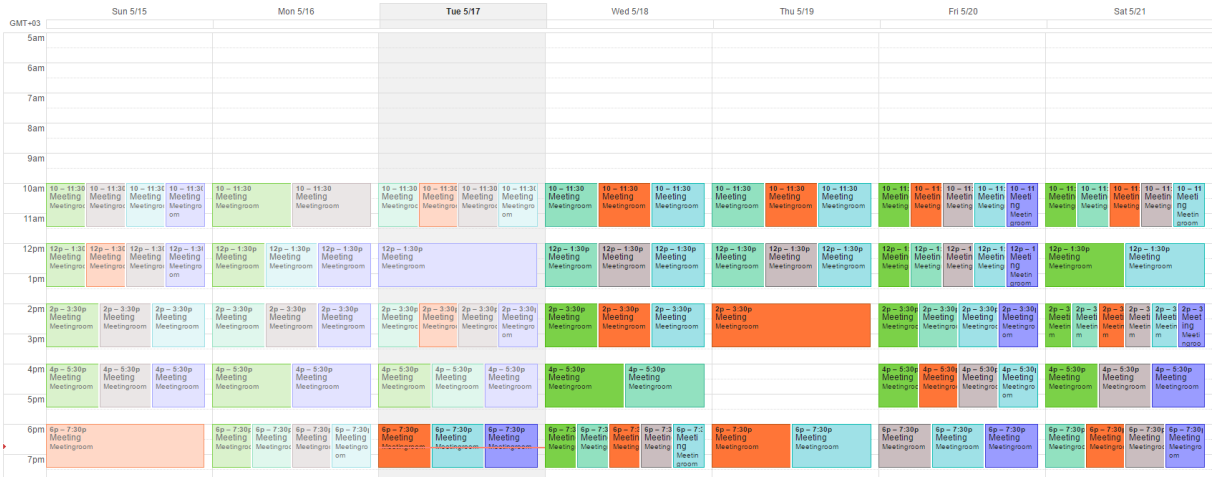
```
String exitTemplate = "$day.0$i.2016|Start $starth:$startm:00.000|End
$endh:$endm:00.000|$room|ID$d|$status|$people|tv=$tv|projector=$projector|v
ideoconf=$videoconf";
```

Joonis 10 - Koosoleku lõpu logikirje šabloon

Joonisel 9 märgitud logikirje šabloon erineb joonisel 10 kujutatud šabloonist selle võrra, et viimasel on ka kirjas andmed selle kohta mis seadmeid koosoleku ajal kasutati. Test andmete puhul on see väärtus juhuslik kuid päris toote juures oleks kasutusel eelduste peatükis mainitud sensorid.

Andmete genereerimise keerulisem osa oli aga need test andmed saada Google Calendar keskkonda. Keskmiselt tekitatakse kirjed umbes 1500 koosoleku jaoks üle kolme kuise ajaperioodi. Käsitsi oleks nende lisamine Google Calendar keskkonda võtnud umbes 4 tundi, arvestades et ühe koosoleku lisamine võtab umbkaudselt 10 sekundit. Õnneks on Google Calendar-il korralik API ning see on võrdlemisi hästi dokumenteeritud. Nii said koosolekud

lisatud vaid minutitega. Joonisel 11 on kujutatud ühe nädala raames loodud koosolekud kõikide koosolekutubade jaoks. Andmete genereerimise osas tekkis ka selline probleem, et kui lisada koosolekutele osalejaid siis programm lakkas töötamast kuna isegi kui e-maili aadressit millega osaleja lisati ei eksisteerinud, tahtis Google ikkagi sellele saata kutse. Pärast kümnenda meili saatmist keeras Google piltlikult öeldes kraanid kinni ning koosolekuid enam ei lisatud. Seetõttu ei saadud defineerida kes täpselt koosolekule olid kutsutud, kuid välise parameetrina sai lisatud koosolekule kutsutute arv.



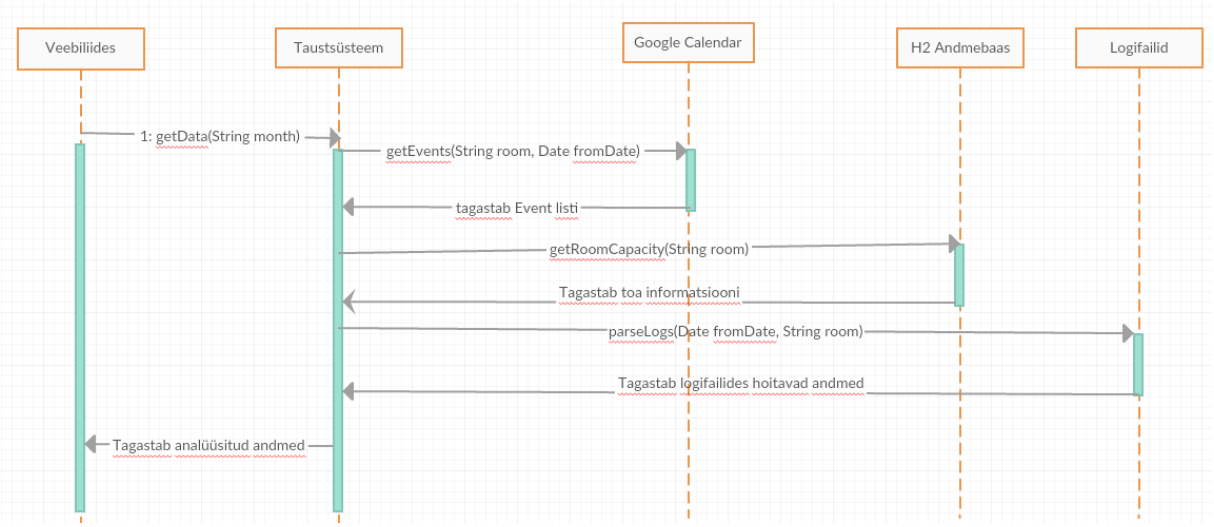
Joonis 11 - Google Calendari genereeritud koosolekud

4.2 Andmete lugemine

Google Calendar keskkonnast andmete kätte saamiseks kasutati Google API-t. Nimelt eelmainitud keskkonnas on koosolekud märgitud kui sündmused, inglise keeles events. Selleks, et kõiki sündmuseid korraga kätte saada kasutati *Maven*-i sõltuvusena projekti lisatud moodulit, mis omakorda teeb HTTP GET päringu aadressile https://www.googleapis.com/calendar/v3/calendars/calendarId/events kus calendarId vahetatakse välja vastava koosolekutoale kuuluva viitega [6].

Saadud vastusest tekitati Java objekt milles leidub kõikide koosolekute algus- ning lõppkuupäev ning kellaaeg, koosolekutele kutsutud inimeste arv, ning viide sellele mis toas koosolek toimus. See info kombineeritakse lokaalses andmebaasis oleva koosoleku toa mahutavusega ning sellega mis seadmed koosoleku toas on. Kui koosolekutoa olemusest on ülevaade olemas, lisatakse infohulgale logifailidest informatsioon selle kohta mitu inimest reaalselt igal koosolekul viibis, kas oli hilinemisi, ning mis seadmeid koosoleku ajal kasutati.

Need andmed tehakse uueks objektiks ning saadetakse JSON formaadis kasutajaliidesele, kus Javascript-i teegid võtavad töö üle ning kuvavad analüüsitud andmeid kasutajale. Arvestades, et nõuetes oli kirjas, et näitama peab andmeid viimase kolme kuu kohta siis saabuvad andmed kolmes osas, iga kuu jaoks üks hulk andmeid. Nii ei koormata liidest üle ning on võimalik andmeid asünkroonselt laadida. See muudab kasutajaliidese kuvamise kiiremaks ning toote kasutaja ei pea väga kaua ootama, et koosolekutoa statistikat näha. Joonisel 12 on kujutatud andmete laadimise jadadiagramm, kust on visualiseeritud kuidas andmeid laadimise protsess.



Joonis 12 - Andmete laadimise jadadiagramm

Lõpuks genereeritakse kasutajale järgnevad andmed:

Joonis 13 - Andmete kuvamise lehekülg

5. Kokkuvõte

Töö põhieesmärkideks oli luua rakendus mis loeb logifailidest andmeid, suudab neid andmeid internetist kättesaadavate andmetega kombineerida ning kombineeritud andmete põhjal analüüsida andmetes olevate koosolekutubade kasutust ning lõpuks kuvada andmete analüüsi tulemusi kliendile kasutajasõbralikus veebiliideses. Samuti oli projekti eesmärk rakendada agiilse arenduse põhimõtteid ning lean startup metoodikat.

Töö tulemusena sai ära analüüsitud projekti vajadused ning valmis ka esimene MVP ehk minimaalne kasutatav toode. Agiilsuse põhimõtetele lähtudes, saab seda jooksvalt täiendada ning toodet on lihtne edasi arendada.

Demontreerisin oma projekti ka Nortal AS personaliosakonnale ning tundus, et huvi oli toote vastu olemas. Lean startup metoodikale lähtudes saaks selle toote ümber luua ka päris äri ning oleks mõistlik tootele luua edasised MVP-d ning toote keerukust kasvatada jooksvalt vastavalt vajadusele.

Võimalikud edasiarendused oleks logifailidele ligipääs üle SSH protokolli ning üldiselt automatiseerida koosolekutubade loomist. Kindlasti saaks ka optimeerida programmi töö käiku sellisel juhul, kuna siis kui andmeid logifailide host masinast rakendusse kantakse saaks analüüsiv osa rakendusest oma tööd juba alustada.

Võib öelda, et kuigi esimeses MVP-s kliendi mugavusfaktor võis jääda veidi puudulikuks, saavutati kindlasti projekti enda eesmärk. Tekkis töötav toode mis suudab logifailides ning Google Calendar keskkonnas olevaid andmeid muuta kasutajale arusaadavamateks graafikuteks ning oskab anda soovitusi kuidas koosolekutubasid efektiivsemalt kasutada. Tootel on kindlasti palju rohkem potentsiaali kui esimeses MVP-s sai arendatud. Kindlasti tuleks vaeva näha, et uut funktsionaalsust lisada ning kasutajale teha toote kasutamine võimalikult lihtsaks ning mugavaks.

Tulemuseks oli töötav rakendus mis suudab andmeid nii genereerida kui ka loodud andmeid lugeda ning nendest luua visuaalselt arusaadavad järeldused. Probleemide lahenduseks loodi algoritmid mis aitasid kaasa toote valmimisele ning selle edukale töökäigule. Leiti alternatiivid puuduolevatele süsteemidele ning suudeti läbi nende API-de edukalt integreerida projektivälised süsteemid. Projekti skoopi arvesse võttes suudeti luua eesmärkidele vastav lõpptoode mis pakkus huvi ka potentsiaalsetele klientidele. Toote modulaarne arhitektuur ning

disain võimaldavad toote lihtsat edasiarendust ning selle mugav ning atraktiivne kasutajaliides on tulevastele klientidele kindlasti meelepärane.

Projekti eesmärk ei olnud luua täiusliku rakendust. Kindlasti on valminud tootel omad puudujäägid ning see on täiesti aktsepteeritav. Lahendused mis eesmärgiks püstitati tehti piisavalt lihtsalt ning piisavalt hästi, et nende paremaks muutmine võtaks sügavamat analüüsi. Andmete loomiseks kirjutati algoritm mis etteantud šablooni põhjal tekitab etteantud ajaperioodi jaoks andmeid nii logifailide näol kui ka Google Calendar keskkonnas hoitavaid koosolekute andmeid.

Lõputöö võib autori poolest nimetada edukaks, kuna toote loomise käigus koguti uut informatsiooni olemasolevate tehnoloogiate kohta ning suudeti neid rakendada toote valmistamisel.

Summary

The main goal of the thesis was to create an application that can read data from log files, can combine that data with data residing in third party web resources, and can analyze meeting room usage based on the combined data and show the results of the analysis to the end user. Another goal was to apply agile development principles and lean startup methodologies.

As a result of the thesis the project's requirements were analyzed and the first MVP, or the minimum viable product, was created. As the project was based on agile principles, continuous integration methods can be applied to develop the product further.

I also demonstrated the product to the human resources department of Nortal and it seemed that there was a potential interest in the product. Based on the methodology defined by lean startup it would be possible to create a sustainable business around the product and it would be reasonable to continue development and create new MPV-s, increasing the product's functionality and complexity as needed.

Possible new developments would include accessing log files over the SSH protocol and to automate meeting room generation in general. It's also possible to optimize the software's processes in that case, as during the file transfer from the log file host machine, the application could simultaneously start analyzing the data it already has access to.

It could be said that even though the customer's comfort factor might be a bit lacking in the first MVP, the project's goal was definitely achieved. A working product was created that can parse log files and data in the Google Calendar environment. It can also generate user-friendly graphs and reports based on the analyzed data and can give recommendations on how to use a company's meeting rooms more efficiently. The product definitely has more potential that was developed in the first MVP. More effort should be put in to add new functionality to the product and to make using the product even simpler and more comfortable.

The result of the thesis was a working application that can both generate log files and data it needs to function and it can also parse the generated data and can create visually pleasing graphs and reports based on the data. To solve the issues that became apparent during the development, algorithms were created to help with the successful functioning of the product. Alternatives were found for systems that were either too complex to use or didn't exist at all and third party systems were successfully integrated through the use of their APIs. Taking the project's scope

into consideration, a successful product was created that meets the requirements and which also was interesting for potential clients. The modular architecture and design of the code allow fir easy future developments and its simple and attractive user interface hopefully suits future customers.

The purpose of the thesis was not to create a perfect application. Certainly the product that was created has its specific deficiencies and shortcomings and that's perfectly acceptable. The result that was created was made simply enough and well enough that finding improvements would certainly need deeper analysis of the issue and the code. To solve the problem of lacking data to analyze, an algorithm was created that generated log files as well as Google Calendar meetings based on a template provided and for a specified time period.

To summarize the author considers the thesis a success, as during the creation of the product new information was gathered on existing technologies and they were successfully integrated to the product during the development of the project.

Kasutatud kirjandus

[1] [Võrgumaterjal]. Available: http://theleanstartup.com/principles. [Kasutatud 14 May 2016].

[2] E. Ries, The Lean Startup, New York: Crown Publishing, 2011.

[3] Agile Estonia, [Võrgumaterjal]. Available: http://agilemanifesto.org/iso/et/principles.html. [Kasutatud 2 May 2016].

[4] Agile Alliance, „What is Agile Software Development?," 3 June 2013. [Võrgumaterjal]. Available: http://www.agilealliance.org/the-alliance/what-is-agile/. [Kasutatud 2 May 2016].

[5] „Bootstrap," [Võrgumaterjal]. Available: http://getbootstrap.com/css/#grid. [Kasutatud 16 April 2016].

[6] [Võrgumaterjal]. Available: https://developers.google.com/google-apps/calendar/v3/reference/events/list#request. [Kasutatud 7 May 2016].

[7] P. Bejger, „Goyello Blog," 21 January 2013. [Võrgumaterjal]. Available: http://blog.goyello.com/2013/01/21/top-9-principles-clean-code/. [Kasutatud 3 May 2016].

[8] R. C. Martin, „Clean Code," %1 *A Handbook of Agile Software Craftsmanship*, Massachusetts, Prentice Hall, 2013, p. 7.

Lisa 1 – Logifailid

```
1.05.2016|Start 9:50:00.000|End 10:10:00.000|milkyway|ID0|Enter|42
1.05.2016|Start 11:20:00.000|End
11:40:00.000|milkyway|ID0|Exit|42|tv=false|projector=false|videoconf=false
1.05.2016|Start 17:50:00.000|End 18:10:00.000|milkyway|ID4|Enter|42
1.05.2016|Start 19:20:00.000|End
19:40:00.000|milkyway|ID4|Exit|42|tv=false|projector=false|videoconf=false
2.05.2016|Start 13:50:00.000|End 14:10:00.000|milkyway|ID7|Enter|49
2.05.2016|Start 15:20:00.000|End
15:40:00.000|milkyway|ID7|Exit|49|tv=false|projector=false|videoconf=false
2.05.2016|Start 17:50:00.000|End 18:10:00.000|milkyway|ID9|Enter|35
2.05.2016|Start 19:20:00.000|End
19:40:00.000|milkyway|ID9|Exit|35|tv=false|projector=true|videoconf=false
3.05.2016|Start 9:50:00.000|End 10:10:00.000|milkyway|ID10|Enter|21
3.05.2016|Start 11:20:00.000|End
11:40:00.000|milkyway|ID10|Exit|21|tv=false|projector=true|videoconf=false
3.05.2016|Start 11:50:00.000|End 12:10:00.000|milkyway|ID11|Enter|16
3.05.2016|Start 13:20:00.000|End
13:40:00.000|milkyway|ID11|Exit|16|tv=false|projector=true|videoconf=false
3.05.2016|Start 17:50:00.000|End 18:10:00.000|milkyway|ID14|Enter|48
3.05.2016|Start 19:20:00.000|End
19:40:00.000|milkyway|ID14|Exit|48|tv=true|projector=true|videoconf=false
4.05.2016|Start 11:50:00.000|End 12:10:00.000|milkyway|ID16|Enter|9
4.05.2016|Start 13:20:00.000|End
13:40:00.000|milkyway|ID16|Exit|9|tv=true|projector=false|videoconf=false
4.05.2016|Start 15:50:00.000|End 16:10:00.000|milkyway|ID18|Enter|28
4.05.2016|Start 17:20:00.000|End
17:40:00.000|milkyway|ID18|Exit|28|tv=false|projector=true|videoconf=false
4.05.2016|Start 17:50:00.000|End 18:10:00.000|milkyway|ID19|Enter|10
4.05.2016|Start 19:20:00.000|End
19:40:00.000|milkyway|ID19|Exit|10|tv=true|projector=true|videoconf=true
5.05.2016|Start 11:50:00.000|End 12:10:00.000|milkyway|ID21|Enter|43
5.05.2016|Start 13:20:00.000|End
13:40:00.000|milkyway|ID21|Exit|43|tv=false|projector=true|videoconf=false
5.05.2016|Start 15:50:00.000|End 16:10:00.000|milkyway|ID23|Enter|22
5.05.2016|Start 17:20:00.000|End
17:40:00.000|milkyway|ID23|Exit|22|tv=false|projector=false|videoconf=false
5.05.2016|Start 17:50:00.000|End 18:10:00.000|milkyway|ID24|Enter|16
5.05.2016|Start 19:20:00.000|End
19:40:00.000|milkyway|ID24|Exit|16|tv=false|projector=true|videoconf=false
6.05.2016|Start 9:50:00.000|End 10:10:00.000|milkyway|ID25|Enter|19
6.05.2016|Start 11:20:00.000|End
11:40:00.000|milkyway|ID25|Exit|19|tv=false|projector=false|videoconf=false
6.05.2016|Start 13:50:00.000|End 14:10:00.000|milkyway|ID27|Enter|50
6.05.2016|Start 15:20:00.000|End
15:40:00.000|milkyway|ID27|Exit|50|tv=false|projector=false|videoconf=false
6.05.2016|Start 15:50:00.000|End 16:10:00.000|milkyway|ID28|Enter|2
6.05.2016|Start 17:20:00.000|End
17:40:00.000|milkyway|ID28|Exit|2|tv=true|projector=true|videoconf=false
7.05.2016|Start 11:50:00.000|End 12:10:00.000|milkyway|ID31|Enter|12
7.05.2016|Start 13:20:00.000|End
13:40:00.000|milkyway|ID31|Exit|12|tv=false|projector=false|videoconf=false
7.05.2016|Start 13:50:00.000|End 14:10:00.000|milkyway|ID32|Enter|3
7.05.2016|Start 15:20:00.000|End
15:40:00.000|milkyway|ID32|Exit|3|tv=false|projector=true|videoconf=false
```

Lisa 2 – H2 SQL Andmebaasi Tabelite Laused

```
DROP TABLE IF EXISTS PUBLIC.Company;
CREATE TABLE PUBLIC.Company(ID INT PRIMARY KEY, NAME VARCHAR(255));

DROP TABLE IF EXISTS PUBLIC.MeetingRoom;
CREATE TABLE PUBLIC.MeetingRoom(ID INT PRIMARY KEY, NAME VARCHAR(255),
CAPACITY INT, COMPANYID INT);
ALTER TABLE PUBLIC.MeetingRoom ADD FOREIGN KEY (COMPANYID) REFERENCES
PUBLIC.COMPANY(ID);

DROP TABLE IF EXISTS PUBLIC.Device;
CREATE TABLE PUBLIC.Device(ID INT PRIMARY KEY, NAME VARCHAR(255), GLYPHICON
VARCHAR(255));

DROP TABLE IF EXISTS PUBLIC.Users;
CREATE TABLE PUBLIC.Users(ID INT PRIMARY KEY, USERNAME VARCHAR(255),
PASSWORD VARCHAR(255), EMAIL VARCHAR(255), ROLE VARCHAR(255));

DROP TABLE IF EXISTS PUBLIC.UserCompanies;
CREATE TABLE PUBLIC.UserCompanies(USERID INT, COMPANYID INT);
ALTER TABLE PUBLIC.UserCompanies ADD FOREIGN KEY (COMPANYID) REFERENCES
PUBLIC.Company(ID);
ALTER TABLE PUBLIC.UserCompanies ADD FOREIGN KEY (USERID) REFERENCES
PUBLIC.Users(ID);

DROP TABLE IF EXISTS PUBLIC.MeetingRoomDevices;
CREATE TABLE PUBLIC.MeetingRoomDevices(MEETINGROOMID INT, DEVICEID INT);
ALTER TABLE PUBLIC.MeetingRoomDevices ADD FOREIGN KEY (MEETINGROOMID)
REFERENCES PUBLIC.MeetingRoom(ID);
ALTER TABLE PUBLIC.MeetingRoomDevices ADD FOREIGN KEY (DEVICEID) REFERENCES
PUBLIC.Device(ID);

DROP TABLE IF EXISTS PUBLIC.Logfile;
CREATE TABLE PUBLIC.Logfile(ID INT PRIMARY KEY, MEETINGROOMID INT, FILENAME
VARCHAR(255), LOGFILE BLOB);
ALTER TABLE PUBLIC.Logfile ADD FOREIGN KEY (MEETINGROOMID) REFERENCES
PUBLIC.MEETINGROOM(ID);
```

Lisa 3 – Viide projekti lähtekoodile

Projekti lähtekoodile saab ligi avalikust githubi repositooriumist aadressil:

https://github.com/mkaar/RoomAnalyzer