TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Ingmar Roos 120420 IABB

VEEBIPÕHISE ÜHELEHERAKENDUSENA TOIMIVA SÕIDUPILETITE KASSAMÜÜGITARKVARA PROTOTÜÜPIMINE

Bakalaureusetöö

Juhendaja: Kadri Umbleja
Tehnikateaduse doktor

Tallinn 2017

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: Ingmar Roos 22.05.2017

Annotatsioon

Käesolevas lõputöös analüüsitakse T grupp AS poolt kasutatava kaugbussiliinide sõidupiletite kassamüügitarkvara teenuspõhisele arhitektuurile üleviimisega seotud probleemistikku. Töö koostamine oli ajendatud vajadusest arendada 2017. aasta jooksul välja ettevõtte olulise tegevusvaldkonna jätkamiseks uus töövahend – veebipõhine kassamüügirakendus – ning koostada selle arenduse sujuvaks ja kuluefektiivseks läbiviimiseks äripoole vajadusi täpselt kirjeldav lähteülesanne.

Töö koostamise käigus kaardistati seni kasutusel olnud kassamüügitarkvara puudused, mis väljenduvad nii süsteemi arhitektuurist tingitud jõudluse probleemides, tarkvara halvas kasutatavuses kui ka ärivaldkonna uutest vajadustest tingitud funktsionaalses mahajäämuses. Töö koostamise **eesmärgiks** oli olemasolevale tarkvarale omaste probleemidele lahendamine ning lahenduse täpse realisatsiooni kirjeldamine reaalse süsteemi toimimist imiteeriva prototüübi kujul.

Töö raames kirjeldas autor loodavale tarkvarale esitatavad nõuded, töötas välja loodava tarkvara kasutajaliidese, kasutatavuse põhimõtted ning viis läbi kasutatavuse testimise loodava süsteemi tulevaste kasutajate peal. Samuti on käesolevas töös uuritud veebipõhiste üheleherakenduste loomiseks sobilikke arendusraamistikke ning esitatud ettepanek loodava tarkvara arendamiseks kasutatavate tehnoloogiate ning komponentide osas.

Töö **tulemusena** on valminud sõidupiletite kassamüügiks ettenähtud veebipõhise üheleherakenduse prototüüp, mis kirjeldab detailselt ja täielikult loodava rakenduse funktsionaalsust ning kasutatavust.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 82 leheküljel, 8 peatükki ning 12 joonist.

Abstract

Prototyping Single Page Application for Public Transport Ticket Sales

The subject of this thesis is analysis of issues related to transferring point of sale (POS) software used by T grupp AS (TG) for long distance bus ticket sales to service based software architecture. The motive for composing this work was a necessity to carry out development of new a new software tool – web based POS software - within year 2017 and to prepare precise description of business requirements as a starting point for this development. This tool is needed to continuously run company's substantial business field – ticket sales through ticket offices.

The shortages of POS software currently in use were mapped in the process of compiling this thesis. These shortages are reflected in performance issues caused by inadequate system architecture, poor usability as well as functional deficiencies which have arisen due to emergence of new business needs. The **purpose** of this thesis was to solve problems which are prone to currently used POS software and to decribe the needed solution in full detail in form of prototype which is imitating the behaviour of the system to be implemented.

In the framework of this thesis the author has described requirements of a software to be created, developed a user interface concept, defined usability principles and conducted usability testing on prospective users. Also, a research on suitable frontend frameworks for creating single page applications was conducted. Based on that, the author has submitted its vision on technologies and components to be used for the development of the intended software.

As a **result** of current thesis a prototype of singe page application for public transport ticket sales offices was created. This prototype describes fully and in detail the functionality as well as the usability of the application to be developed.

The thesis is in Estonian and contains 82 pages of text, 8 chapters, 12 figures.

Lühendite ja mõistete sõnastik

AJAX	Asynchronous JavaScript and XML, asünkroonse serveripäringu tehnika
API	Application Programming Interface, rakendusliides ehk programmliides
ASCII	American Standard Code for Information Interchange, Ameerika standardne koodisüsteem infovahetuseks
Baassoodustus	Mingile kindlale sõitjagrupile kehtestatud sõidusoodustus, mille alusel müüdavate piletite arv ei ole piiratud. Näiteks eelkooliealise sõidusoodustus.
EMV	Europay, MasterCard, and Visa
CSS	Cascading Style Sheets, veebilehe kujundusdefinitsioonide kaskaadlaadistik
DOM	Document Object Model, dokumendi objektimudel. Platvormist ja keelest sõltumatu XML, XHTML ja HTML dokumentidega suhtlemise liides.
HTTP	Hypertext Transfer Protocol, hüperteksti ülekandeprotokoll
HTTPS	Hypertext Transfer Protocol Secure, turvaline hüperteksti ülekandeprotokoll
JSON	JavaScript Object Notation, andmevahetusvorming, mis põhineb JavaScripti programmeerimiskeele alamhulgal
Kampaaniakood	Tpileti müügisüsteemis kasutatav aktiveerimiskood, mille abil on võimalik rakendada kindla aja jooksul või kindlale arvule piletite soodustusi.
Liin	Kindlate väljumisaegade ja kindla marsruudiga regulaarne ühistransporditeenus.
Lisateenus	Piletimüügisüsteemis müüdav igasugune teenus või toode, mis ei ole sõidupilet.
KMR	Kassamüügirakendus sõidupiletite kassamüügiks kasutatav tarkvara

MVC	Model-View-Controller. Arhitektuur, mis jagab objektid kolmeks (mudel, vaade, kontroller) ja määrab ära, kuidas need omavahel suhtlevad.
Müügikanal	Füüsiline või virtuaalne kanal, milles teostatakse piletimüüki.
Müügikvoot	Bussifirma poolt Tpileti süsteemis müümiseks eraldatud piletite kogus.
Müügipunkt	Kindel koht või virtuaalse keskkonna ilming, kus teostatakse piletimüüki (näiteks Tallinna bussijaama 2. kassa või tpilet.ee veebikauplus).
Ostukorv	Ühe ostusessiooni raames müüdavate piletite ja lisateenuse kogum.
PINS kaart	Air Balticu lojaalsussüsteemist välja arenenud kaubandusettevõtete ülese lojaalsusprogrammi kliendikaart.
POS	Point of Sale, müügipunkt
Reis	Kindlal kuupäeval ja väljumisajal osutatav bussiveoteenus. Reisi teenindatakse kindla ühissõidukiga, mistõttu on reisi maksimaalne sõitjate arv piiratud ühissõiduki sõitjakohtade arvuga.
REST	Representational State Transfer, andmevahetusteenuste tehnoloogia
Sooduskaart	Bussifirma või piletimüügiagendi poolt aktsepteeritav kliendikaart, mille esitaja saab pileti ostmisel hinnasoodustust.
TCP/IP	Transmission Control Protocol/Internet Protocol, kirjelduslik raamistik arvutivõrgu protokolli jaoks.
TG	T grupp AS
URI	Uniform Resource Identifier, üldine ressursiidentifikaator
WCAG	World Wide Web Consortiumi (W3C) poolt välja töötatud veebilehtede juurdepääsetavuse juhendmaterjalid (ingl. Web Content Accessibility Guidelines)
WCF	Windows Communication Foundation
Wireframe	Ekraanivaate visand, mis kirjeldab ligikaudselt elementide paigutust ja nende toimimist.

Sisukord

Jooniste loend

1. Sissejuhatus

Käesolev lõputöö käsitleb ühistranspordi tugiteenuste ettevõtte T grupp AS (*TG*) piletimüügi ärivaldkonna, täpseni sõidupiletite kassamüügi tarkvara analüüsi. Käsitletav probleem seisneb selles, et ettevõttes seni kasutusel olnud kassamüügirakendus (*KMR*) ei vasta ärivaldkonna kasvanud vajadustele, samuti on süsteemi andmestruktuuride keerukuse ja tehingute arvu kasvu tõttu tekkinud arvestatavad jõudlusprobleemid KMR-i kasutamisel. Jõudlusprobleemi tuumikpõhjuse lahendamiseks on ettevõte algatanud enda infosüsteemi ulatusliku muudatuse, mille tulemusena juurutatakse teenuspõhine arhitektuur. Selle muudatuse taustal on ettevõtte juhtkond otsustanud piletite kassamüügi teostamiseks välja töötada uue veebipõhise KMR-i, mille abil lahendatakse lisaks jõudlusprobleemile ka muud hetkel kasutusel olevale tarkvarale omased probleemid.

Lõputöö ülesandeks on töötada välja uue veebipõhise KMR-i arendamiseks detailne lähteülesanne, mis kirjeldab täpselt loodavale tarkvarale esitatavaid nõudeid nii funktsionaalsuse kui ka kasutatavuse osas. Lähteülesande koostamise meetodiks on valitud prototüüpimisel põhinev analüüsimudel, mille rakendamise eesmärgiks on vältida spetsifitseerimisele keskenduvale analüüsiprotsessile omaseid puudusi, mis väljenduvad lähteülesande mitmeti mõistetavuses ning sellest tingitud vajadusest teha juba valmis arendatud tootes muudatusi.

Hästi õnnestunud spetsiaaltarkvara üheks oluliseks eelduseks on äriprotsessidest aru saamine ning äripoole probleemide ja vajaduste mõistmine. Töö esimene pool (peatükid 2, 3 ja 4) keskendub TG ärimudeli kirjeldamisele, hetkel kasutusel oleva tarkvara puuduste kaardistamisele ning uue KMR-i nõuete määratlemisele. Töö teises osas (peatükid 5 ja 6) on kirjeldatud eelnevalt kaardistatud nõuete põhjal uue tarkvara toimimist detailselt imiteeriva prototüübi loomist. Eraldi on keskendutud loodava KMR-i kasutatavuse analüüsimisele ning selle käigus viidud läbi prototüübi kasutatavuse testimisele, milles osalesid süsteemi tulevased kasutajad.

Töö viimases osas uuritake veebipõhiste üheleherakenduste loomiseks sobilikke arendusvahendeid ning kaardistatakse lahendused, kuidas lõimida veebipõhise tarkvaraga piletimüügi teostamiseks vajaminevad riistvaraseadmed.

Töö tulemusena on valminud arendatava KMR-i toimimist kogu funktsionaalsuse ulatuses kirjeldav prototüüp, mis on 2017. aasta teises pooles läbiviidava arenduse lähteülesandeks.

2. Organisatsioon ja ärimudel

2.1. Organisatsiooni taust

TG on 1998. aastal loodud ettevõte, mille peamisteks tegevusaladeks on läbi tegutsemisperioodi olnud ühistranspordisektori ettevõtetele piletimüügiteenuse pakkumine ja bussijaamade opereerimine. TG kuulub rohkem kui 1000 töötajaga Mootor Grupp AS-i koosseisu. Ettevõtte asutamisest kuni 2010. aastani kandis äriühing nime Bussireisid OÜ. Sel tegutsemisperioodil kujunes äriühing Eesti olulisimaks bussipiletite müügiga tegelevaks ettevõtteks ning arendas *bussireisid.ee* veebikaupluse näol välja sõidupiletite internetimüügi valdkonna, mida enne seda Eestis praktiliselt ei eksisteerinud. Esimese 12 aasta vältel oli firma oluliseks tegevusalaks ka bussidega tellimusvedude teostamine.

2010. aastal viidi läbi ärinime muudatus ning ettevõte omandas uue ärinime Tallinna Bussijaam OÜ. Formaalse muudatuse taustal kujundas ettevõte ümber ka enda tegevusvaldkonnad ning lõpetas tellimusvedude teostamise. See oli ajendatud eesmärgist keskenduda bussiettevõtetele füüsilise ja digitaalse taristuteenuse pakkumisele ning lõpetada tellimusvedude teostajana konkureerimine samade bussiettevõtetega, kellele taristuteenust pakuti. Ühtlasi uuendati ettevõtte piletimüügivaldkonna identiteeti koondades kõik piletimüügiteenustega seotud tooted ja teenused kaubamärgi Tpilet alla.

Juba neli aastat hiljem, 2014. aastal omandas ettevõte taas uue ärinime, milleks sai T grupp AS. Ka seekord oli formaalne muudatus ajendatud ärifookuse ümberkujundamisest mõlemas ärivaldkonnas. Esmalt tekkis seni üksnes Tallinna Bussijaama operaatorina tegutseval ettevõttel ambitsioon ja reaalne äriperspektiiv kujuneda Eesti olulisemate bussijaamade võrgustiku operaatoriks, mistõttu ei kirjeldanud ärinimi Tallinna Bussijaam OÜ enam tegevusala geograafilist ulatust. 2015. aastal asus ettevõte opereerima Tartu ja Rakvere bussijaamasid ning läbi 2012. aastal omandatud tütarettevõtte Cargobus OÜ tegutseti veel üheksas maakonnakeskuse bussijaamas. Teisalt tekkis ettevõtte juhtkonnal ja omanikel äratundmine, et algselt üksnes Eesti turul tegutsevatele bussiettevõtete piletimüügiteenuse pakkumiseks loodud IT süsteem on oma arengukõveras jõudnud

etappi, kus on tekkinud arvestatav äriperspektiiv selle turundamiseks Euroopa bussiettevõtetele tarkvaratootena – T solutions piletimüügiplatvormina.

Alates 2014. aastast on ettevõtte tegevust iseloomustanud tarkvara arendamise valdkonna kiire kasv ning äri geograafilise ulatuse laienemine Eestist väljapoole. 2016. aastaks omandas tarkvaraarenduse valdkond niivõrd suure mastaabi, et omanike otsusega eraldati see ärivaldkond TG kooseisust ning moodustati eraldi äriühing T solutions OÜ, mis tegutseb alates 2017. aastast TG tütarettevõttena. TG jätkab äritegevust kahes tuumikvaldkonnas - bussijaamade võrgustiku opereerimine ning piletimüügi- ja infoteenuste pakkumine nii füüsilises kui ka virtuaalses keskkonnas. Neis ärivaldkondades toetavad emaettevõtte tegevust mõlemad tütarettevõtted – Cargobus OÜ esinduste kaudu on laiendatud füüsiliste kanalite piletimüük bussijaamadesse, mille eraldiseisev opereerimine ei oleks tasuv ning T solutions OÜ tagab elektroonilise piletimüügiplatvormi järjepideva arendamise.

2.2. Ärimudeli kirjeldus ja areng

Käesolev lõputöö keskendub piletimüügiteenuste ärivaldkonnale, mistõttu on siinkohal esitatud üksnes vastava ärimudeli üldine kirjeldus.

TG osutab kaubamärgi Tpilet all piletimüügiteenuseid rohkem kui 20-le kodu- ja välismaisele bussiettevõttele. Piletimüüki teostatakse nii füüsilistes müügipunktides, milleks on bussijaamade kassad ja bussijaamades paikneva piletimüügiautomaadid kui ka virtuaalsetes keskkondades, milleks on tpilet.ee veebikauplus ja Tpileti mobiilirakendus. Vahetult enda personaliga teostab TG piletimüüki üksnes Tallinna ja Tartu bussijaamas. Üheksas väiksemas bussijaamas teostatakse piletimüüki all-agendina tegutseva tütarettevõtte Cargobus OÜ vahendusel. Lisaks sellele müüakse kolmes Eesti bussijaamas (Viljandis, Pärnus ja Kuressaares) Tpileti süsteemi vahendusel pileteid frantsiisi- ja tarkvara litsentsilepingute alusel. See tähendab, et vastavates bussijaamades piletimüüki teostavad ettevõtted on müüdavate piletite osas vahetus lepingulises suhtes bussiettevõtetega, kuid kasutavad müügiinstumendina Tpileti tarkvara ning teostavad piletimüüki Tpileti kaubamärgi teatud ühtsete teeninduspõhimõtete alusel.

Lõpptarbijale (bussireisijale) pakutavateks teenuse ja brändi väärtusteks on valikuvabadus, lihtsus, usaldusväärsus. Need väärtused väljenduvad selles, et Tpileti

kaudu on bussireisijal võimalik osta praktiliselt kõikide Eesti bussiettevõtete kaugliinide sõidupileteid, pileti ostmise protsess on kõikides müügikanalites lihtne ning Tpileti kaudu ostetud piletiga on reisijale alati bussis koht tagatud, samas kui vahetult enne reisi bussijuhilt ostes võib buss olla välja müüdud. Nende lõpptarbijatele pakutavate väärtuste ning suure füüsilise müügivõrgustiku (bussijaamade) kaudu genereerib Tpilet bussiettevõtetele olulist osa reisijate koguarvust, mistõttu on bussiettevõtted motiveeritud enda teenindavate bussiliinide pileteid Tpileti kanalites müümiseks eraldama.

Ärimudeli kohaselt teenib TG tulu müüdavatele piletitele kohaldatud vahendustasust ehk müügiprovisjonist. See tähendab, et Tpileti müügikanalitest müüdud piletite müügikäibest kohustub vedaja maksma agendilepingus kindlaksmääratud protsendi TG-le kui piletimüügiteenuse osutajale. Kirjeldatud ärimudeli tugevuseks on asjaolu, et bussireisija jaoks ei ole mistahes Tpileti müügikanalist pileti ostmisel pileti hind kunagi kallim enne reisi väljumist bussijuhilt ostes. Bussiettevõtete jaoks seondub samas Tpileti müügikanalite kasutamine teatava kuluga, mistõttu on oluline, et Tpileti müügiteenus pakuks bussiettevõtetele piisavat lisandväärtust selle kulu põhjendamiseks.

Kuivõrd kõik vedajad müüvad sõidupileteid paralleelselt Tpileti müügikanalitega ka ise, on agendilepingutega määratletud nii müüki antavate piletite arv iga reisi lõikes, kui ka eelmüügi avamise aeg, kasutatavad müügikanalid ja müügipunktid, piletimüügi sulgemise aeg ning piletite tagastamise ning sellega seonduvate teenustasude kohaldamise reeglid. Enamlevinud põhimõtte kohaselt käivitatakse piletimüük 10 päeva enne reisi väljumist ning internetimüük lõpetatakse 1 tund ja kassamüük 5 minutit enne bussi väljumist algpeatusest. Kirjeldatud näide on aga üksnes turuliidrite eeskujul välja kujundatud *de facto* standard, mis on viimase kolme aasta jooksul hakanud bussiettevõtete lõikes oluliselt varieeruma. Samuti on oluliselt muutunud piletite hinnastamise põhimõtted. Nimetatud muutus väljendub ennekõike selles, et kindlaksmääratud staatilise hinnakirja asemel rakendavad mitmed bussiettevõtted nõudluse ja eelmüügiaja põhiseid dünaamilisi hinnakirjasid, ning erinevate turundusinstrumentidega seotud kindla sihtgrupi soodustusi. Bussireisija jaoks väljendub see muutus asjaolus, et mida varem enne reisi väljumisaega pilet ostetakse, seda soodsam hind kehtib. Tpileti müügikanalite tehnilises vaates lisab see aga oluliselt keerukust, kuivõrd kõikide kanalite eelmüügil tuleb arvestada hinna muutumisega ning tuvastada hetkel kehtiv pileti hind.

Sarnaselt e-kaubanduse levikule muudes ärivaldkondades on ka TG tegevusalale iseloomulik virtuaalsete müügikanalite osakaalu kasv ning füüsiliste müügipunktide populaarsuse vähenemine. Kui 2014. aastal müüdi 76,6% eelmüügipiletitest läbi füüsiliste müügipunktide (piletikassad ja automaadid), siis 2016. aastal oli sama näitaja kahanenud 60,9%-ni. Samal ajal on populaarsust kogunud ka mitme bussiettevõtte enda veebimüügikeskkonnad, mistõttu on bussijaamades paiknevatest füüsilistest müügipunktidest müüdavate piletite osakaalu langus olnud tegelikkuses veel suurem. Sellele vaatamata ei ole kassade näol tegemist kaugeltki marginaalse tähtsusega kanaliga, mistõttu on oluline tagada bussijaamade kassamüügi sujuv toimimine ning rakendada selleks töökindlat tarkvara, mille funktsionaalsus vastab ärivaldkonna vajadustele.

2.3. Probleemi kirjeldus

Olemasolev KMR on loodud ja kasutusele võetud 2011. aastal arvestades piletimüügivaldkonna toonaseid ärivajadusi. Kasutusaja jooksul toimunud jätkuarenduste ja regulaarsete versiooniuuenduste käigus on tarkvara funktsionaalsust täiendatud, kuid sellele vaatamata on ärivaldkonna arengu käigus kerkinud üles teatud funktsionaalseid vajakajäämisi, mille tõttu ei ole kassamüügil võimalik teostada osasid operatsioone, mis looksid kliendile (bussireisijale) arvestatavat lisaväärtust. Näiteks on klienditeenindajal võimalik müüa pileteid küll kindla numbriga istekohale, kuid talle ei ole nähtav bussi istmeplaan, mistõttu ei ole pileti müümisel võimalik arvestada kliendi sooviga istekoha paiknemise kohta bussis. Samuti ei ole võimalik müüdud piletite andmeid muuta. Seega on praeguses olukorras kassamüügil võimalik vedajate piletimüügi äriprotsesse järgida üksnes lihtsustatud kujul.

Teiseks kriitiliseks probleemiks on KMR-i halb jõudlus, mille tõttu ei ole tippajal võimalik müüa pileteid vajaliku kiirusega ning seetõttu kannatab nii klienditeeninduse kvaliteet kui ka ettevõtte tulu. Olemasoleva tarkvara loomisel püstitas ettevõtte juhtkond nõude, mille kohaselt pidi üksikpileti müümise protsess alates sihtkoha otsingust kuni makseviisi valimiseni olema vilunud kasutajal läbitav mitte rohkem kui 5 sekundiga. Tänases olukorras võtab ainuüksi otsingukriteeriumidele vastavate reiside leidmine 5-10 sekundit ning etaloniks seatud protsess võtab keskmiselt 20 sekundit.

Kirjeldatud jõudluse probleem on tingitud andmebaasiserveri ning klientprogrammi teenindava andmevahetuskihti realiseeriva serverirakenduse halvast jõudlusest, mille on

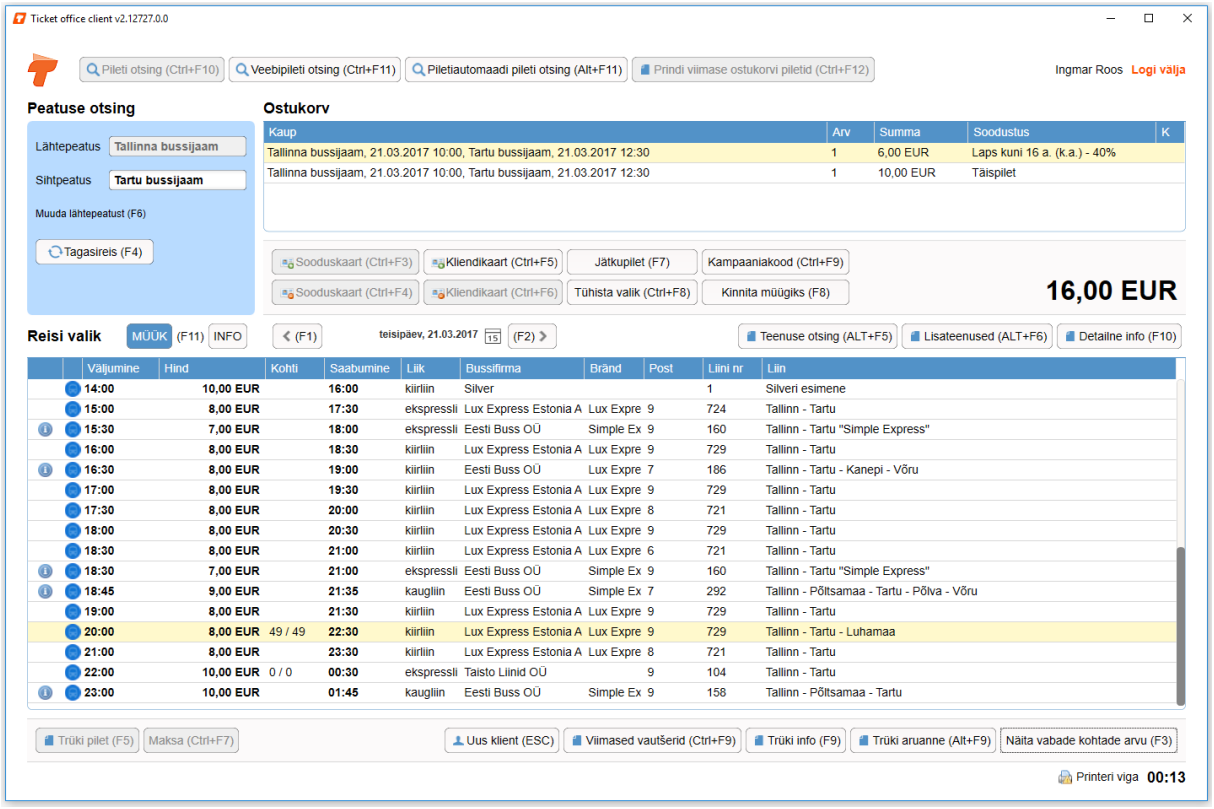
kaasa toonud andmemudeli keerukuse kasv. Arendusmeeskond on jõudnud veendumusele, et praeguseks on ammendunud võimalused kompenseerida seda keerukuse kasvu päringute optimeerimise ja serveriressursi suurendamise teel. Seetõttu on alustatud ulatusliku andmemudeli ja süsteemi arhitektuuri parendusega, mille tulemusena juurutatakse süsteemi teenuspõhine arhitektuur – st võetakse süsteemi keskserveri ja kõikide klientrakenduste vaheliseks suhtluseks kasutusele REST *(Representational State Transfer)* veebiteenustel põhinev andmevahetuskiht. Selle muudatuse elluviimisel ei ole olemasolev KMR-i senisel kujul enam kasutatav.

Kokkuvõtvalt avaldub probleemi olemus selles, et T solutions tervikliku piletimüügiplatvormi jõudlusprobleemide lahendamiseks on vaja oluliselt muuta süsteemi arhitektuuri ja andmemudelit. Selle käigus muudetakse ka süsteemi komponentide vahelist andmevahetuskihti, millest tingituna tuleks hetkel kasutusel oleva KMR-i jätkuvaks funktsioneerimiseks oluliselt muuta selle komponendi ja serverirakenduse vahelist andmevahetuse lahendust. Kuivõrd hetkel kasutusel olevas KMR-s on avaldunud ka mitmeid funktsionaalseid puudujääke, ei oleks aga üksnes andmevahetuskihi muutmiseks tarvilike arenduste läbiviimine pikemaajalisi äriperspektiive arvesse võttes jätkusuutlik lahendus. Olemasoleva tarkvara funktsionaalsuse parandamine koos andmevahetuse poole muutmisega tooks kaasa aga niivõrd suures mahus arenduse, et majanduslikult on mõistlikum töötada välja uus KMR.

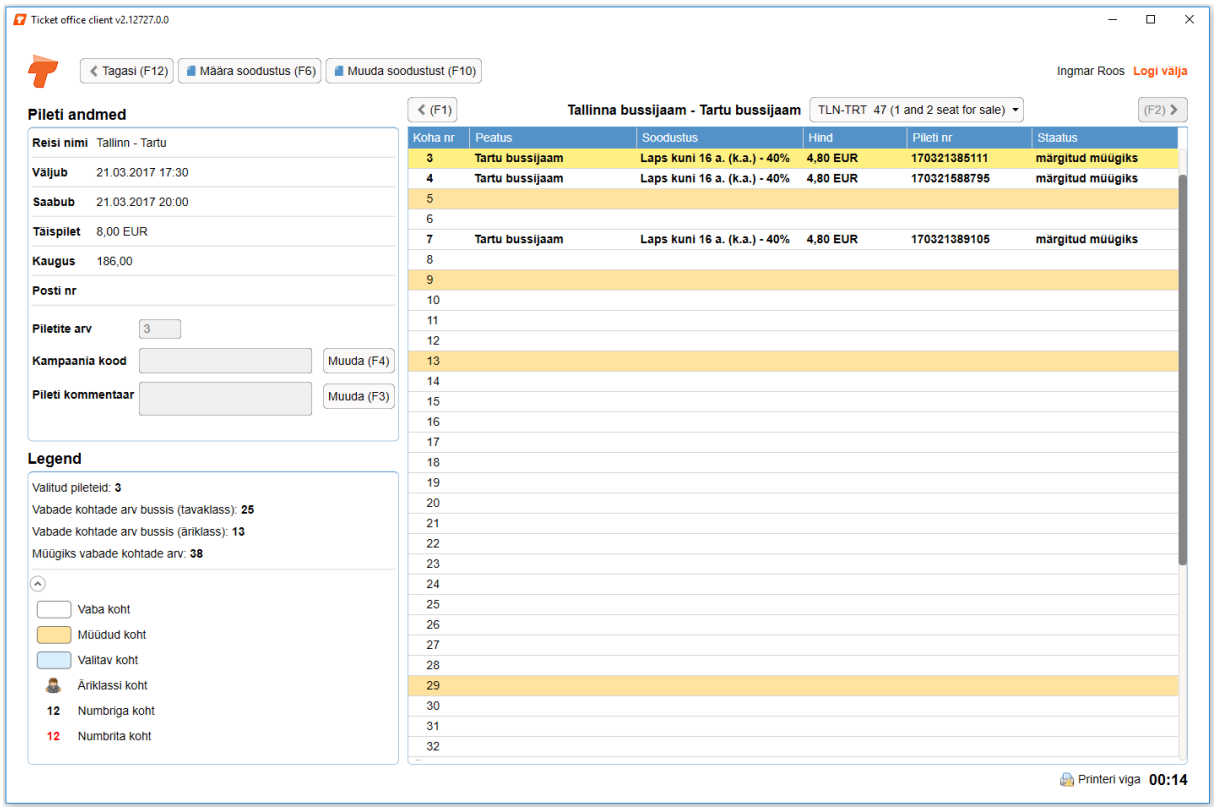
3. Olemasoleva tarkvara kirjeldus

Olemasoleva KMR-i näol on tegemist .NET platvormil arendatud *desktop* rakendusega, mis on ette nähtud piletimüügisüsteemis kirjeldatud bussiliinide sõiduplaanide kohta klientidele info andmiseks ning nendele liinidele sõidupiletite müümiseks. Kogu piletimüügiks vajamineva andmestiku haldus toimub veebipõhises serverirakenduses, mida loetakse terviksüsteemi mõistes eraldi komponendiks ning mida käesolevas töös põhjalikumalt ei käsitleta. Seega on KMR-i näol tegemist klientprogrammiga, mis suhtleb andmebaasiga läbi serverirakenduse kasutades selleks WCF *(Windows Communication Foundation)* andmevahetusraamistikku

Olemasoleva tarkvara *desktop* rakendusena juurutamise peamiseks põhjuseks oli vajadus tagada piletimüügiprotsessi maksimaalne kiirus, mille üheks eelduseks oli see, et kõiki põhiprotsesse on võimalik läbida kasutades selleks klaviatuuri kiirkäske. Alljärgnevalt on esitatud ekraanitõmmised olemasoleva KMR-i kasutajaliidesest:



Joonis 1. KMR-i kasutajaliides. Reiside otsingu ekraanivorm.



Joonis 2. KMR-i kasutajaliides. Pileti lisamise ekraanivorm

Teine oluline argument KMR-i *desktop* rakendusena juurutamiseks oli perifeersete riistvaraseadmete lõimimise lihtsus. Nendeks seadmeteks on piletite termoprinter ja kaardimakseterminal, mille olemasolu on piletite kassamüügil vältimatu. Võrreldes veebirakendustega on operatsioonisüsteemile paigaldatud rakenduse koodist tarkvaratehniliselt oluliselt vahetum ligipääs spetsiaalriistvara juhtimiseks kasutatavatele füüsilistele portidele ning riistavaraseadmete juhtimiseks ettenähtud draiveritele.

Tavapäraselt on *desktop* rakenduste arendamine eelistatud lahendus ka juhtudel, kui ei ole võimalik tagada stabiilse internetiühenduse olemasolu [1]. Siiski ei ole TG äriprotsesside spetsiifikast tulenevalt võimalik *desktop* rakenduse abil saavutada täiendavat toimekindlust võrguühenduse ajutise katkemise korral. Põhjuseks on see, et T solutions piletimüügiplatvormil toimub piletimüük paralleelselt erinevates müügipunktides ning müügikanalites (sh piletikassad, veebikauplused, piletiautomaadid, bussi käsimüügiterminalid), mistõttu tuleb paralleelprotsessidest tingitud konfliktide vältimiseks kontrollida iga transaktsiooni eel vabade kohtade arvu, valitud istekoha saadavust, hetkel kehtivat piletihinda jms.

Kasutusel oleva KMR-i desktop rakendusena juurutamise valiku selgitamisel tuleb arvestada ka asjaoluga, et 2011. aastal, mil hetkel kasutuses olev KMR välja töötati, ei olnud veebipõhiste rakenduste loomiseks kasutatavad tehnoloogiad nagu veebiserveri poolel töödeldav Javascript ning vastava tehnoloogia rakendamist hõlbustavad arendusraamistikud veel sellise küpsusega, et suurt jõudlust ja töökiirust eeldavate töökohtade realiseerimine veebibrauseris käitavate üheleherakendustena oleks olnud levinud lahendus.

3.1. Olemasoleva tarkvara puudused

Olemasoleval KMR-l on mitmed vajakajäämised, mis jagunevad funktsionaalseteks puudusteks, kasutatavust mõjutavateks puudusteks ning mittefunktsionaalseteks puudusteks. Käesoleva töö raames loodava uue KMR-i prototüübi puhul on eesmärgiks likvideerida kasutajaliidesest alltoodud puudused.

3.1.1. Funktsionaalsed puudused

- Kasutajal ei ole võimalik müüdud piletite andmeid muuta. Pileti muutmine on osade bussivedajate poolt kehtestatud piletimüügitingimuste kohaselt lubatud toiming, mida tuleb klienditeeninduse huvides reisijale võimaldada. Näiteks peab olema võimalik muuta piletile märgitud algpeatust juhul, kui kehtivat piletit omav reisija teatab, et ta soovib peale tulla mitte piletile märgitud algpeatusest vaid mõnest sellele järgnevast peatusest. Samuti peab programm võimaldama muuta reisija soovil piletile märgitud istekohta.
- Kasutajal ei ole piisavalt infot reisijale kindla istekoha müümiseks. Olemasolev KMR võimaldab küll kindla numbriga istekohtade müümist, kuid kasutajaliides ei anna kasutajale infot selle numbriga istekoha paiknemise kohta bussis. Seetõttu ei ole klienditeenindajal võimalik piletimüügil arvestada kliendi soovidega müüa talle näiteks istmerea aknapoolne koht või müüa kahele koos reisivale kliendile kõrvuti asetsevad istekohad.
- Rakendus ei tagasta reisi otsingul ümberistumistega reise. Mitmetesse sihtkohtadesse reisimiseks on vaja kasutada kahte või enamat bussiliini. Praegu peab sellistel juhtumitel klienditeenindaja enda teadmiste piires kombineerima kokku kliendi poolt küsitud reisi, otsides ükshaaval reise, mille vahelise ümberistumisega on võimalik sõita soovitud sihtkohta.

- Piletiga ei ole võimalik siduda kõiki vajaminevaid andmed, mis on nõutavad rahvusvaheliste liinide piletimüügil. Sellisteks andmeteks on reisija telefoni number, e-posti aadress, dokumendi number ja sugu. Osa nimetatud andmete kogumine piletimüügil on nõutav Vene Föderatsioonis kehtiva seadusandluse alusel ning seetõttu ei ole olemasolevat KMR-i võimalik kasutada Venemaale suunduvate rahvusvaheliste liinide piletimüügiks ka siis, kui bussivedajad enda nõuetes järeleandmisi teeksid.
- KMR-s ei ole esitatud infot bussi mugavusvarustuse kohta. Erinevate alternatiivide olemasolul langetab reisija enda otsuse kasutatava bussiliini osas võttes arvesse bussis olemasolevaid mugavusi (näiteks Wifi, WC ja salongi meediakeskuste olemasolu). Omamata infot vastavate mugavuste olemasolu kohta, ei ole klienditeenindajal võimalik anda reisijale kogu teavet ostuotsuse langetamiseks.

3.1.2. Kasutatavuse puudused

- Piletimüügiprotsessi läbimiseks vajaminev klahvivajutuste arv ei ole mõistlik ning pärsib müügi kiirust. Kasutaja peab piletimüügiprotsessi põhivoo läbimiseks navigeerima reiside otsingu ekraanivormi (joonis 1) ja pileti valiku ekraanivormi (joonis 2) vahel ning tegema klahvivajutusi nende vormide aktiveerimiseks ja deaktiveerimiseks.
- Müüdavate reiside vabade kohtade arvu nägemiseks tuleb klienditeenindajal vabade kohtade arv iga reisi kaupa manuaalselt aktiveerida. Tippnõudlusega aegadel, mil suur osa reisidest müüakse välja, on selline protsess eriti tülikas, kuna kasutaja peab sobiva reisi leidmiseks käima ükshaaval läbi kõik reisid kuni vabade kohtadega reisi leidmiseni.
- Reisija andmete sisestamisel tuleb aktiveerida ja täita sisestusväljad iga ostukorvis oleva pileti kaupa eraldi ning kinnitada selle piletiga seotud andmed enne järgmise juurde asumist. Selline protsess raiskab kahe või enama reisija korral piletimüügiks kuluvat aega ning kahjustab klienditeeninduse efektiivsust.
- Kasutajale ei ole soodustusega piletihinnad nähtavad enne täishinnaga pileti ostukorvi asetamist ja sellele seejärel soodustuse rakendamist. Sageli soovib soodustuse õigusega reisija saada infot erinevate vedajate poolt pakutavate

sooduspileti hindade kohta ning nendele päringutele vastamiseks peab klienditeenindaja teostama sama operatsiooni korduvalt.

3.1.3. Mittefunktsionaalsed puudused

- Rakenduse toimimise kiirus ei ole rahuldav. KMR-i toimimise kiirus on järk-järgult halvenenud tingituna piletimüügisüsteemi andmemudeli keerukuse kasvust. Jõudlusprobleem on tingitud olemasoleva andmevahetuskihi halvast jõudlusest, mis on olulisim põhjus miks T solutions piletimüügiplatvormi arhitektuuri ja andmemudeli muudatused ette võetakse. Seega ei ole tegemist probleemiga, mida oleks võimalik lahendada KMR-i poolel, kuid koos teenuspõhise arhitektuuri juurutamisega laheneb ka kirjeldatud probleem.
- Desktop rakenduse seadete konfigureerimine toimub kasutaja arvuti failisüsteemi tasemel. Teatud juhtudel on see raskendatud, sest rakendust litsentsilepingu alusel kasutavatel osapooltel ei ole seadistamiseks piisavat pädevust. Samas ei ole andmeturbe põhimõtete või tehniliste võimaluste puudumise tõttu võimalik TG administraatorile anda ka kaughaldusega juurdepääsu teise osapoole arvutisse.

4. Uue tarkvara nõuded

4.1. Allsüsteemi eesmärgid ja kasutajad

Arendatav KMR on T solutions piletimüügiplatvormi üks allsüsteem, mille loomise ja kasutuselevõtmise eesmärgiks on tagada infotehnoloogine töövahend bussijaamade klienditeeninduse protsesside läbiviimiseks. Sama eesmärki täidab ka käesoleval ajal kasutusel olev KMR, kuid tingituna äriliste vajaduste muutumisest on tarkvara kaasabil läbiviidavate protsesside hulk ning sisu mõnevõrra muutunud. KMR-i abil läbiviidavate protsesside loend on esitatud punkti 4.2 all ning kõiki neid protsesse teostab klienditeenindaja. Seega on tarkvara näol tegemist spetsiifiliselt klienditeenindaja töökohal kasutamiseks ettenähtud töövahendiga ning muid kasutajaid selle allsüsteemi puhul ei eksisteeri. Seejuures ei ole vaadeldava süsteemi vaates isegi administraatori rolli kuivõrd KMR-i kasutajate haldus ning õiguste omistamine toimub terviksüsteemi kontekstis teises komponendis – andmehalduse allsüsteemis. Arvestades asjaolu, et kõikide KMR-i abil läbiviidavate protsesside puhul on tarkvaraga suhtlevaks inimkasutajaks klienditeenindaja (kes omakorda suhtleb kliendiga suuliselt), ei ole töös esitatud rollipõhist kasutusjuhtude eskiismudelit.

4.2. Põhiprotsessid

Alljärgnevalt on esitatud KMR-i abil teostatavad klienditeenindaja töökoha põhiprotsessid:

- Pileti müümine
- Pileti tühistamine
- Pileti muutmine
- Pileti väljatrükk
- Liiniinfo andmine
- Liiniinfo väljatrüki koostamine
- Lisateenuse müümine

- Aruande koostamine

Põhiprotsessidena on käsitletud puhtalt klienditeeninduse toimimiseks läbiviidavaid tegevusi, millele on üldjuhul iseloomulik klienditeenindaja ja teise isiku (klient, bussijuht) vaheline suhtlus. Seetõttu ei ole põhiprotsessidena loetletud KMR-is tehtavaid administratiivset laadi tegevusi nagu süsteemi sisse ja välja logimine.

4.3. Nõuded tarkvarale

4.3.1. Funktsionaalsed nõuded

- Peab olema võimalik otsida infot kõikide süsteemi sisestatud liinide sõiduplaanide kohta.
- Liinide otsingut peab olema võimalik teostada lähte- ja sihtkoha sisestamise teel.
- Liinide otsingu tulemust peab olema võimalik sõiduplaanina välja trükkida
- Sõiduplaani väljatrükil peab olema esitatud otsingus defineeritud peatuste vaheliste liinide sõiduplaan mõlemas suunas (sinnasuund ja tagasisuund)
- Peab olema võimalik otsida infot kõikide süsteemi sisestatud liinide reiside kohta.
- Reiside otsingut peab olema võimalik teostada lähte ja sihtkoha sisestamise teel.
- Peab olema võimalik määrata reisi toimumise kuupäeva.
- Reiside otsing peab kasutajale tagastama nii otsereiside ühendused kui ka ümberistumistega reiside tulemused.
- Reisi otsingu tulemustes peab olema kasutajale eristatavalt esitatud juba väljunud reisid ning veel väljumata reisid, samuti reisid, millele on võimalik pileteid müüa ja millele mitte.
- Kasutajale peab kuvatama infot müügiks ettenähtud vabade kohtade arvu kohta.
- Kasutajale peab kuvatama infot reisi teenindava bussi mugavusvarustuse kohta.
- Müügiks eraldatud sõitjakohtadega reisidele peab olema võimalik pileteid müüa.
- Peab olema võimalik läbida pileti müümise protsessi üksnes arvuti klaviatuuri kiirkäske kasutades. St kasutajaliideses navigeerimine peab olema võimalik ilma arvuti hiirt kasutamata.
- Pileti müümisel peab ostukorv aeguma kui kasutaja ei teosta maksetehingut süsteemis määratud aja jooksul.
- Pileteid peab olema võimalik välja trükkida
- Pileti väljatrükk peab pileti müümisel toimuma automaatselt.

- Peab olema võimalik müüa lisateenuseid.
- Kui reisil on määratud istmeplaan, siis peab kasutajal olema võimalik pileti müügil määrata piletile vaba istekoht istmeplaanilt valimise teel.
- Peab olema võimalik sisestada iga pileti kohta reisi parameetrites defineeritud andmeid reisija kohta.
- Peab olema võimalik pileteid tühistada.
- Pileti tühistamisel peab olema võimalik määrata tühistamise põhjus.
- Pileti tühistamisel peab kasutajale kuvama kliendile tagastatava summa.
- Peab olema võimalik otsida süsteemist müüdud pileteid pileti numbri, reisija nime, viitenumbri müügikanali ning müümise kuupäeva ja reisi toimumise kuupäeva alusel.
- Müüdud pileteid peab olema võimalik otsida kõikide müügikanalite, müügipunktide ja süsteemi kasutajate ulatuses.
- Otsingu tulemusena leitud piletit peab olema võimalik välja printida. Samuti peab olema võimalik leitud pileti alusel välja printida kõiki teisi sama ostukorvi pileteid
- Peab olema võimalik müüdud piletit muuta. Muuta peab olema võimalik lähtepeatust, reisija andmeid, ning istekohta.
- Peab olema võimalik mitme pileti või lisateenuse müümine ühe maksetehinguga.
- Peab olema võimalik tasuda sularahas, pangakaardiga, kliendikaardiga ning vautšeriga.
- Pangakaardiga maksmisel peab makseterminali summa sisestamine toimuma läbi KMR-i kasutajaliidese (st realiseeritud liides makseterminaliga).
- Vautšeriga tasumisel peab olema võimalik tasuda vautšeri väärtusest puudujääv osa muu makseviisiga.
- Peab olema võimalik registreerida sooduskaarti ja rakendada sellele vastavat soodustust ostukorvile.
- PINS kaardi sooduskaardina registreerimisel peab süsteem automaatselt tuvastama kaardi omaniku nime ning lisama nime reisija andmetesse.
- Peab olema võimalik sisestada kampaaniakoodi ja rakendada kampaaniakoodile vastavat soodustust ostukorvile.
- Peab olema võimalik koostada järgmisi aruandeid: kassaaruanne, tühistatud piletite aruanne, piletite kordustrükkide aruanne ja reisi eelmüügileht.

- Kassaaruannet peab olema võimalik koostada ainult sisseloginud kasutaja poolt tehtud tehingute kohta. Muid aruandeid peab olema võimalik koostada kõikide kasutajate tehingute kohta.
- Tasuliste toimingute (pileti muutmine, liiniinfo väljatrükk, pileti müümine) teenustasud peavad toimingu teostamisel lisanduma ostukorvi maksumusele automaatselt.
- Peab olema võimalik süsteemi sisse ja välja logida kasutajanime ja parooli alusel.
- Ei tohi olla võimalik kasutada KMR-i funktsionaalust ilma süsteemi logimata.
- Kasutajaliides peab olema üles ehitatud selliselt, et aktiivseks on muudetud ainult läbiviidava kasutusjuhu kontekstis relevantsed nupud ja sisestusväljad.
- Kasutajaliidese sisestusväljadel peab toimuma sisestatud väärtuse formaadi proaktiivne valideerimine.
- Kasutajaliidese paneelide fookus peab liikuma kasutusjuhu kontekstis loogilises järjekorras (näiteks mingi valiku kinnitamisel aktiveerub automaatselt paneel, milles tuleb kasutajal teha järgmine valik).
- Kasutajale kuvatavad veateated peavad olema tõrke põhjust selgitava (st ei tohi kasutajaliideses esitada tarkvaraplatvormi sisseehitatud veateate koode ja seletusi).

4.3.2. Mittefunktsionaalsed nõuded:

- Süsteem peab olema kasutatav vähemalt 100 samaaegse kasutaja korral.
- Pileti müümise põhiprotsess (ilma sooduskaarti või kampaaniakoodi rakendamata) peab olema läbitav 5 sekundi jooksul.
- KMR peab olema kasutatav enamlevinud veebibrauseritega ilma klientarvutisse rakenduse spetsiifilist tarkvara paigaldamata (välja arvatud perifeersete riistvaraseadmete draiverid).
- Kõik rakenduses tehtavad toimingud ja nendega seotud andmete päringud peavad toimuma asünkroonselt ilma esitluskihti brauseris uuesti laadimata.
- Lähte- ja sihtkoha otsingus kasutatavad peatuste andmed peavad olema puhverdatud klientarvuti mäluruumi. Peatuste andmete uuendamine toimub rakendusse sisse logimisel.

- Juhul, kui operatsiooni teostamiseks vajaminevate andmete kuvamine võtab kauem kui 0,5 sekundit peab kasutajale kuvama info poolelioleva andmevahetuse kohta.
- Kogu andmevahetus klientarvuti ja serveri vahel peab toimuma HTTPS *(Hypertext Transfer Protocol Secure)* protokolli vahendusel.
- Rakenduse kasutajaliides peab toetama mitmekeelsust, kusjuures keel määratakse vastavalt kasutajakonto seadistusele.
- Kasutajaliideses kuvatavate aktiivsete menüüpaneelide teksti ja tausta kontrastsus peab vastama vähemalt WCAG 2.0 tasemele AA.
- Kasutajaliides peab olema skaleeruv (size responsive) ning olema optimeeritud ekraaniresolutsioonile alates 1280 x 720 px.

4.4. Kasutusjuhtude kirjeldused

Alljärgnevalt on esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis. Kasutusjuhtude kirjeldused selgitavad punktis 4.2 loetletud põhiprotsesside läbiviimist, kuid võttes arvesse, et mõni põhiprotsess hõlmab suurt arvu üksteisele järgnevaid tegevusi ning protsessidel on kasutusjuhtude tasemel ühisosasid, on kasutusjuhtude tasemel jaotatud osad protsessid alamprotsessideks. Kõikide kasutusjuhtude tegutseja on klienditeenindaja, mistõttu ei ole tegutsejat igakordselt märgitud.

Kasutusjuht: Kasutaja identifitseerimine (UC1)

Kirjeldus: Kasutaja identifitseerib ennast sisestades kasutajanime ja parooli. Süsteem autendib kasutaja, st kontrollib kasutaja väidetavat identiteeti. Kui kasutaja identifitseerimine õnnestub, siis lubatakse kasutajal süsteemi siseneda, vastasel juhul mitte.

Kasutusjuht: Pileti ostukorvi lisamine (UC2)

Kirjeldus: Kasutaja sisestab reisi lähte- ja sihtkoha ning reisi toimumise kuupäeva, Süsteem kuvab reiside nimekirja. Kasutaja valib sobiva reisi. Süsteem kuvab kõikide saadaolevate piletite hinnad (täispiletid ja baassoodustusega piletid). Kasutaja määrab müüdavate piletite arvu. Kui valitud reisil on defineeritud bussi istmeplaan, määrab kasutaja piletitele vastavad istekohad. Kasutaja kinnitab valiku. Süsteem tekitab ostukorvi objekti ja märgib sellega seotud piletid seisundisse „märgitud müügiks“. Juhul,

kui tegemist on ümberistumisega reisiga valib kasutaja jätkureisi, määrab piletite arvu ja kinnitab. Süsteem lisab teise (ja järgnevate) reiside piletite kinnitamisel piletid varem loodud ostukorvile.

Kasutusjuht: Lisateenuse ostukorvi lisamine (UC3)

Kirjeldus: Kasutaja aktiveerib lisateenuse valiku paneeli. Süsteem kuvab saadaolevad lisateenused. Kasutaja valib kliendi poolt soovitud lisateenuse ja määrab koguse. Kasutaja kinnitab valiku. Kui eksisteerib aktiivne ostukorv, siis lisab süsteem valitud lisateenuse ostukorvile. Kui ostukorvi ei eksisteeri, siis tekitab süsteem uue ostukorvi objekti.

Kasutusjuht: Ostukorvi muutmine (UC4)

Eeltingimus: Kasutaja on lisanud ostukorvi vähemalt ühe pileti või lisateenuse.
Kirjeldus: Kasutaja valib ostukorvis oleva pileti või lisateenuse, mille ta soovib ostukorvist eemaldada. Süsteem eemaldab pileti või lisateenuse ostukorvist. Juhul, kui kasutaja valib eemaldamiseks viimase ostukorvis oleva pileti või lisateenuse, kustutatakse ka ostukorvi objekt. Kõikide piletite ja lisateenuste korraga ostukorvist eemaldamiseks valib kasutaja „ostukorvi tühistamine“. Süsteem eemaldab kõik piletid ja lisateenused ostukorvist. Süsteem kustutab ostukorvi objekti.

Kasutusjuht: Reisija andmete sisestamine (UC5)

Eeltingimus: Kasutaja on lisanud ostukorvi vähemalt ühe pileti, millel on reisija andmete sisestamine lubatud või kohustuslik.
Kirjeldus: Kasutaja valib menüüvaliku „reisija andmed“. Süsteem kuvab reisija andmete sisestusväljad. Kohustuslikud ja mittekohustuslikud andmeväljad on erinevalt tähistatud. Kasutaja täidab reisija andmete väljad. Süsteem kontrollib kohustuslike andmete olemasolu. Kui kõikidele kohustuslikele väljadele on andmed sisestatud, aktiveerib süsteem valiku „Maksa“.

Kasutusjuht: Soodustuse rakendamine (UC6)

Eeltingimus: Kasutaja on ostukorvi lisanud vähemalt ühe pileti.
Kirjeldus: Kasutaja sisestab kampaaniakoodi ja/või sooduskaardi andmed. Magnetribaga sooduskaardi puhul võib sooduskaardi andmete sisestamine toimuda kaardilugeja abil. Kasutaja kinnitab sisestatud andmed. Süsteem kontrollib kampaaniakoodi ja/või sooduskaardi kehtivust. Kui sisestatud kaardist või kampaaniakoodist tulenev soodustus

kehtib ostukorvis olevatel reisidel, rakendab süsteem ostukorvis olevatele piletitele soodustuse. Vastupidisel juhul annab süsteem veateate kampaaniakoodi või sooduskaardi mitte kehtimise kohta.

Kasutusjuht: Personaalse kliendikaardi rakendamine (UC7)

Eeltingimus: Kasutaja on ostukorvi lisanud vähemalt ühe pileti, millel kehtib personaalne sooduskaart (PINS[1] kaart).

Kirjeldus: Kasutaja vajutab ostukorvi jaotises nupule "Reisija andmed“. Süsteem kuvab reisija andmete jaotise. Kasutaja sisestab PINS kaardi numbri klaviatuurilt või magnetriba lugejaga. Süsteem tuvastab PINS kaardi numbri alusel kaardi omaniku ees- ja perenime ning täidab vastavad väljad ekraanivormil. Juhul, kui kasutaja üritab sisestada sama PINS kaardi numbrit kahele või enamale ostukorvis olevale sama reisi piletile, annab süsteem veateate selgitusega, et kliendikaart on personaalne.

Kasutusjuht: Ostukorvi tasumine (UC8)

Eeltingimus: Kasutaja on lisanud ostukorvi vähemalt ühe pileti või lisateenuse.

Kirjeldus: Kasutaja vajutab ekraaninupule „Maksa“. Süsteem kuvab võimalikud makseviisid: sularaha, pangakaart, kliendikaart ja vautšer. Kasutaja valib kliendi poolt eelistatud makseviisi. Kliendikaardi või vautšeriga maksmise korral sisestab kasutaja kaardi või vautšeri numbri. Kasutaja kinnitab makse. Süsteem trükib välja pileti (juhul, kui ostukorvi oli pilet lisatud). Juhul, kui klient tasub vautšeriga, mille jääkväärtus on väiksem kui ostukorvi maksumus, siis tuleb ülejäänud osa tasumiseks tegevust korrata tasudes puudujääv osa muu makseviisi või teise vautšeriga.

Kasutusjuht: Liiniinfo otsimine (UC9)

Kirjeldus: Kasutaja aktiveerib menüüjaotise „Info“. Süsteem kuvab lähte- ja sihtkoha sisestusväljad. Kui eelnevalt oli teostatud reisi otsing, siis käivitab süsteem liini sõiduplaanide otsingu nende alusel. Muul juhul sisestab kasutaja lähte- ja sihtkoha. Süsteem kuvab lähte- ja sihtkoha vaheliste liinide sõiduplaani koondvaate, millel on toodud info reiside käigusoleku nädalapäevade kohta. Süsteem aktiveerib ekraaninupu „Liiniinfo väljatrükk“.

[1] Lihtsustuse huvides on kasutusjuhu kirjelduses viidatud PINS kaardile, mis oli töö kirjutamise ajal ainuke Tpileti süsteemis kasutav personaliseeritud sooduskaardi liik. Praktikas võivad bussiettevõtted kasutusele võtta ka muid personalseid sooduskaarte.

Kasutusjuht: Liiniinfo trükkimine (UC10)

Eeltingimus: Kasutaja on teostanud liiniinfo otsingu

Kirjeldus: Kasutaja vajutab ekraaninupule „Liiniinfo väljatrükk“. Süsteem trükib välja sõiduplaani koondvaate, millel on toodud kõikide nädalapäevade reisid lähte- ja sihtkoha vahel ning vastupidi. Süsteem lisab ostukorvi teenustasu. Kasutaja vajutab ekraaninupule „Maksa“ ja teostab makse vastavalt kasutusjuhu „Ostukorvi tasumine“ kirjeldusele.

Kasutusjuht: Pileti otsimine (UC11)

Kirjeldus: Kasutaja aktiveerib pileti otsingu vaate. Kasutaja sisestab pileti otsimiseks vähemalt ühe järgnevatest: a) pileti number b) viitenumber c) reisija nimi ja ostmise kuupäev d) reisija nimi ja reisi kuupäev. Kohustuslikele andmetele lisaks võib kasutaja otsingu täpsustamiseks määrata müügikanali. Juhul, kui müügikanaliks on määratud „piletiautomaat“ saab otsingu sisendparameetrites defineerida mitu minutit tagasi pilet osteti. Süsteem kuvab otsinguparameetritele vastavate piletite nimekirja. Tulemustes sisalduva pileti kirjele klõpsates kuvab süsteem pileti detailandmed.

Kasutusjuht: Pileti muutmine (UC12)

Eeltingimus: Kasutaja on teostanud pileti otsingu. Otsingu tulemusena leitud pilet on ärireeglite kohaselt muudetav.

Kirjeldus: Kasutaja vajutab pileti detailandmete vaates nupule „Muuda“. Süsteem kuvab ekraanivormi, millel muudetavad sisestusväljad on aktiivsed ja mittemuudetavad väljad mitteaktiivsed. Kasutaja muudab vajaminevaid andmeid ja valib „Kinnita“. Süsteem annab hoiatuse selle kohta, et peale muutmist ei ole esialgne pilet enam kehtiv. Kasutaja kinnitab muudatuse. Süsteem lisab ostukorvi pileti muutmise teenuse. Kasutaja vajutab „Maksa“ ja teostab makse vastavalt kasutusjuhu „Ostukorvi tasumine“ kirjeldusele. Süsteem trükib välja muudetud pileti.

Kasutusjuht: Pileti tühistamine (UC13)

Eeltingimus: Kasutaja on teostanud pileti otsingu.

Kirjeldus: Kasutaja vajutab pileti detailandmete vaates nupule „Tühista“. Süsteem kuvab ekraanivormi, tagastamiseks vajalike andmete väljadega. Kasutaja määrab tühistamise põhjuse ja sisestab kommentaari. Süsteem arvutab kliendile tagastatava summa. Kui tagastamise põhjus on „valesti müüdud“, siis on kasutajal võimalik tagastatavat summat muuta. Kasutaja kinnitab tagastuse. Süsteem trükib tagastuse kviitungi.

Kasutusjuht: Pileti trükkimine (UC14)

Eeltingimus: Kasutaja on teostanud pileti otsingu.
Kirjeldus: Kasutaja vajutab pileti detailandmete vaates nupule „Trüki“. Süsteem trükib valitud pileti. Süsteem lisab väljatrükkimise teenuse ostukorvi. Kasutaja vajutab „Maksa“ ja teostab makse vastavalt kasutusjuhu „Ostukorvi tasumine“ kirjeldusele.

Kasutusjuht: Aruande koostamine (UC15)

Kirjeldus: Kasutaja valib menüüjaotisest „Aruanded“ vajamineva aruande. Süsteem avab brauseris uue vahelehe ja suunab kasutaja Tpileti aruandlusmooduli keskkonda. Aruandlusmoodulis on eeltäidetud valitud aruande sisendparameetrite väljad. Kasutaja käivitab aruande koostamise. Süsteem kuvab aruande ekraanil. Kasutaja valib aruande eksportimise soovitud formaati (xls, csv või pdf). Süsteem konverteerib aruande valitud formaati.

Kasutusjuht: Eelmüügilehe koostamine (UC16)

Kirjeldus: Kasutaja teostab reisi otsingu sisestades lähte- ja sihtkoha ning reisi kuupäeva (vaikimisi hetke kuupäev). Süsteem kuvab otsingutulemustele vastavate reiside loendi. Kasutaja aktiveerib nimekirjas reisi. Kasutaja valib menüüjaotisest „Aruanded“ ja valib eelmüügilehe aruande. Süsteem laadib Tpileti aruandlusmooduli keskkonnast alla eelmüügilehe pdf kujul. Kasutaja avab alla laaditud aruande ja trükib selle vajadusel välja.

Kasutusjuht: Süsteemist väljalogimine (UC17)

Kirjeldus: Kasutaja vajutab ekraanivormil kirjele „Logi välja“. Juhul, kui rakenduses on pooleliolev ostukorv annab süsteem hoiatuse, et väljalogimisel pooleliolev ostukorv tühistatakse. Kasutaja kinnitab, et soovib sellele vaatamata välja logida. Süsteem logib kasutaja välja ning kuvab süsteemi sisselogimise ekraanivormi.

4.5. Ärireeglid

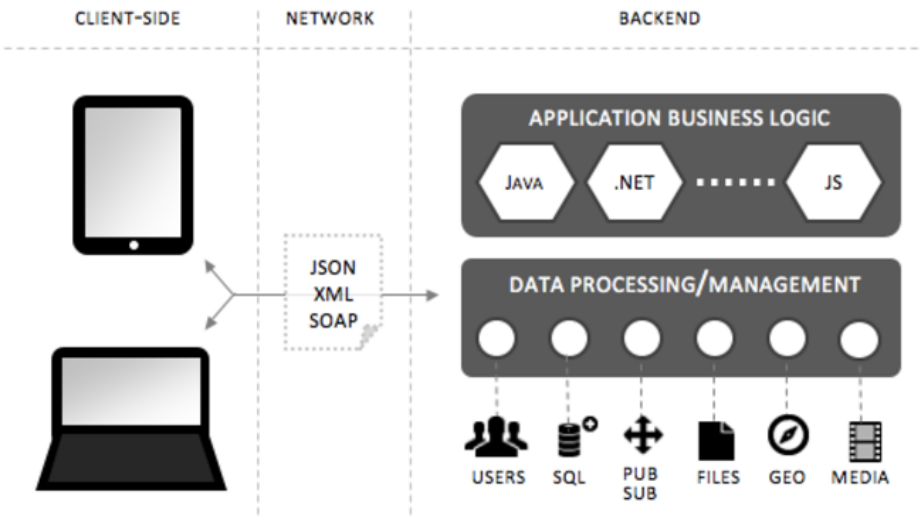
Loodava KMR-i näol on tegemist klientrakendusega, mis kasutab ja töötleb andmeid läbi REST veebiteenustel põhineva Tpileti API *(Application Programming Interface)*. Neid veebiteenuseid kasutavad lisaks KMR-le ka muud rakendused, näiteks Tpileti mobiilirakendus. Perspektiivis võib Tpileti API-t hakata kasutama ka kolmandate osapoolte tarkvarad. Sellises olukorras kerkib klassikaline teenuspõhise arhitektuuri

küsimus – millises ulatuses tasub ärireegleid jõustada klientrakenduse poolel ning millises osas peab ärireeglid jõustama andmevahetusteenuseid realiseerivas äriloogika kihis ehk *backend* süsteemis.

Üldiselt kehtib rusikareegel, et kõik klientrakenduste üleselt kehtivad ärireeglid tuleb jõustada (vähemalt) äriloogika kihis. Sellisel lähenemisel on mitmeid eelised. Esiteks on see oluline turvakaalutlustel. Kui klientrakenduse näol on tegemist eessüsteemi (edaspidi *frontend)* rakendusega siis on selles jõustatud ärireeglitest võimalik mööda hiilida ja süstida süsteemiga manipuleerimise eesmärgil ärireeglitega vastuolus olevaid andmekomplekte.

Klientrakenduste poolel jõustatud ärireeglitega seotud oht ei pruugi olla alati seotud pahatahtliku manipuleerimisega, vaid ka asjaoluga, et erinevate klientrakenduste arendajad võivad ärireeglitest erinevalt aru saada ja neid erinevalt või mittetäielikult jõustada.

Ärireeglite äriloogika kihis jõustamise eeliseks on ka asjaolu, et kui ärireegleid jõustatakse ja hallatakse ühes kohas, siis elimineerib see reeglite muutumise korral vajaduse sünkroniseerida reeglite muudatusi klientrakendustes. Ärireeglite muutumise korral on võimalik teha muudatus üksnes äriloogika kihis ja see muudatus kandub üle API vahendusel äriloogikat „tarbivatele“ klientrakendustele. [2]



Joonis 3 Äriloogika ja esitluskihi eraldamine teenuspõhise arhitektuuri puhul [3]

Samas, juhul kui kõik süsteemiga seotud ärireeglid jõustada ainult äriloogika kihis, siis on väga keeruline luua mõistliku kasutatavusega klientrakendust või põhjustaks see ebamõistlikult suure koormuse API-le. Näiteks oleks üsna totter lasta KMR-s kasutajal sisestada piletite koguseks negatiivseid väärtuseid ja jätta selle situatsiooni lahendamine äriloogika kihile.

Ilmselt ei ole klientrakenduse iga sisestusvälja väärtuse valideerimiseks mõistlik pöörduda andmevahetusteenuse poole. Seetõttu on praktikas parem realiseerida staatilisena püsivate ärireeglite ulatuses *frontend* poolel teatav liiasus, mitte pöörduda igakordselt API poole sisestatud väärtuse kontrollimiseks. [4]

Seega taandus loodava KMR-i puhul jõustamist vajavate ärireeglite defineerimine küsimusele, milliste ärireeglite „vahetu kättesaadavus“ on vajalik, et realiseerida sisestusväljade väärtuste proaktiivset valideerimist ning minimaalsete nõutavate andmekomplektide olemasolu kontrolli. Vajaminevad ärireeglid tuvastas autor prototüübi ekraanivaadete põhjal, määratledes millised sisestusväljade väärtused ja väärtuste komplektid on T solutions piletimüügiplatvormi äriloogikat arvestades loogilised ja lubatud ning samas piisavalt püsiva iseloomuga, et neid oleks mõistlik klientrakenduse poolel jõustada. KMR-i poolel jõustatavad ärireeglid on järgmised:

- Ühe reisi piires ei saa ostukorvi lisada rohkem pileteid, kui on antud reisile müümiseks eraldatud vabade kohtade arv.
- Väljamüüdud reisidele ei saa pileteid müüa.
- Ostukorvi lisatavate piletite ja/või lisateenuste kogus peab olema nullist erinev täisarv.
- Minevikus toimunud reisile ei saa piletit müüa.
- Minevikus toimunud reisi piletit ei saa muuta.
- Minevikus toimunud reisi piletit saab tühistada ainult tühistamise põhjusega „bussifirma süül“.
- Kohanumbritega reisidel ei saa olla samal marsruudi lõigul üks istekoht seotud rohkem kui ühe piletiga.
- Reisi lähte- ja sihtkoht ei saa olla sama peatus.
- Aegunud ostukorvi ei saa pileteid lisada ega sealt pileteid eemaldada.
- Aegunud ostukorvi eest ei saa tasuda.

- Peale ostukorvi eest tasumist ei saa sellesse pileteid ja lisateenuseid lisada ega eemaldada.
- Kui pilet on seotud PINS kaardi numbriga, peab pilet olema seotud ka reisija ees- ja perekonnanimega.
- Ühe reisi piires ei saa sama PINS kaart olla seotud rohkem kui ühe piletiga.
- Müügikanalitest „veebikauplus“ ja „mobiilirakendus“ ostetud pileteid saab otsida viitenumbri alusel.
- Kõikidest müügikanalitest ostetud pileteid saab otsida pileti numbri alusel.
- Kõikidest müügikanalitest ostetud pileteid saab otsida reisija nime ja ostmise kuupäeva alusel.
- Kõikidest müügikanalitest ostetud pileteid saab otsida reisija nime ja reisi toimumise kuupäeva alusel.
- Müügikanalist „piletimüügiautomaat“ ostetud pileteid saab otsida müügipunkti ja müümise hetkest möödunud aja alusel.
- Pileti muutmisel saab muuta reisija andmeid, lähtepeatust ja istekoha numbrit.
- Pileti lähtepeatuse muutmisel saab määrata uueks lähtepeatuseks ainult selline peatus, mis paikneb reisi marsruudil peale esialgset lähtepeatust ning enne sihtpeatust.

5. Prototüüpimine

5.1. Prototüüpimisel põhinev arendusprotsess

Käesoleva bakalaureusetöö käigus teostatud analüüsiprotsessis kasutas autor prototüüpimisel põhinevat arendusmustrit (ingl. *prototype driven development*). Kuigi tarkvara või selle osade kohta abstraktsete mudelite või visuaalsete sketšide koostamist on arendusprotsessides kasutatud ka varem, olid meetodid nagu *mockuping, wireframing, storyboarding* ja muud tänapäevase tarkvaraarenduse analüüsiga seotud metoodikad ca 15 aastat tagasi veel tundmatud ning on ilmunud koos agiilse tarkvaraarendusega massilisse kasutusse alles viimase 10 aasta jooksul [5].

Prototüüpimisel põhineva arendusmetoodika eeliseks on asjaolu, et visuaalsed prototüübid on lihtsasti arusaadavad kõikidele seotud osapooltele – sh ilma tehniliste teadmisteta osalistele. Traditsioonilisel nõuete spetsifitseerimisele ja dokumenteerimisele keskenduval analüüsil põhineva arendusprotsessi puhul peab klient nähtava tulemi saamiseks ootama nädalaid lootes, et analüüsietapis läbiviidud intervjuude põhjal kirjapandud nõuete pinnalt on sündinud tema vajadustele ja visioonile vastav toode. Seevastu on prototüüpimisele keskenduva analüüsi korral võimalik väga kiiresti anda kliendile hoomatavat tulemit [6].

Autor ei vastanda siiski prototüüpimist tarkvara nõuete korrektsele dokumenteerimisele vaid peab neid teineteist täiendavateks. Uurimisobjektiks oleva kasutajaliidese prototüübi koostamise käigus selgus, et funktsionaalseid nõudeid tuli oluliselt täiendada ning eelnevalt kirjapandud kasutuslugude kirjeldusi muuta. Seejuures ei olnud prototüüpimine mitte täiendavaid nõudeid tekitav vaid vigade ennetamist ning varast avastamist võimaldav tegevus.

Tarkvaraarenduse projektide tüüpiliseks probleemiks on nähtus, kus tarkvara telliv klient ei suuda enda vajadusi kirjeldades hoomata kõiki oma äriprotsesside seisukohalt olulisi nüansse. Teine äärmus on juhtum, kus klient peab mingeid enda ärivaldkonna eripäradest tulenevaid nõudeid niivõrd ilmselgeks, et ei tule selle pealegi, et neid tuleks analüüsi

etapis eraldi välja tuua [5]. Tavapäraselt ei ole aga projekti teostaja poolel tegutsev analüütik tellija valdkonna ekspert, mistõttu ei suuda ta tellija käsitluse järgi ilmselgete nõuete peale tulla.

„*Tarkvaraettevõtete klassikaliseks õudusunenäoks on kliendid, kes spetsiaaltarkvara tellimise käigus hakkavad alles enda äri ja sellega seotud protsesse välja mõtlema ning soovivad seejuures jääda algselt kokkulepitud aja ja eelarve raamidesse*“ [5].

Käesoleva töö puhul oli autor väga hästi kursis äripoole nõuete ja olemasolevates protsessides esinevate kitsaskohtadega. Sellele vaatamata selgus prototüübi loomise ja tarkvara tulevastele kasutajatele tutvustamise käigus mitmeid asjaolusid, mille peale ei olnud autor prototüübi esmase versiooni loomisel tulnud ja mille pinnalt õnnestus ära hoida vajaminevate muudatuste ilmsiks tulemist programmerimise, testimise või juurutuse faasis. Selle põhjalt võib eeldada, et arendusprojektides, kus analüütiku teadmised ärivaldkonna nüanssidest on pinnapealsemad, aitab kasutajaliidese prototüüpimine veelgi tõhusamalt tuvastada äripoole spetsiifilisi nõudeid ning vähendada vajaminevate muudatuste arvu arendustsükli hilisemates faasides.

5.2. Prototüüpimise arendusvahendid

Turul on saadaval lai valik erinevaid kasutajaliideste prototüüpimiseks ettenähtud tarkvaratooteid. Autori eesmärgiks oli luua arendatavale lõpptootele lähedase, detailse graafilise kujundusega täisfunktsionaalne prototüüp[1], millest lähtuvalt toimus ka kasutava tarkvara valik. Valiku tegemisel katsetas autor nelja alternatiivi alljärgnevate prototüüpimise tarkvarade hulgast: Balsamic Mockups, Axure, Justinmind Prototyper ja InVision. Taotletavast eesmärgist lähtuvalt pidas autor oluliseks:

- Kasutajaliidese elementide interaktiivse toimimise defineerimise võimalust (näiteks määrata, kuidas muutub mingi nupu stiil sisestusväljale väärtuse sisestamise järel).
- Graafika täpsust ja detailsust, samuti võimalust üle võtta prototüübis kasutatavaid graafika definitsioone lõpptoote arendamisel.

[1] „Täisfunktsionaalne prototüüp on keskkond, mis võimaldab anda lõppkasutajale reaalset kasutuskogemust loodava tarkvaraga, ilma seda tarkvara valmis programmeerimata“ [5].

- Kasutaja poolt defineeritavate komponentide loomise ja korduvkasutamise võimalust.
- Eeldefineeritud kasutajaliidese elemente sisaldavate teekide importimise võimalust.
- Prototüübi huvigruppidele kättesaadavaks tegemise lihtsust.

Alljärgnevalt on esitatud 7 populaarseima prototüüpimistarkvara omaduste võrdlus UX disainerite blogi prototypr.io koostajate hinnangul[1]:

Prototyping Tools		Mockplus	Axure	Balsamiq	JustInmind	Sketch	Adobe XD (Preview	Invision
Productivity	Learning Curve	Very Easy	Complex	Very Easy	Complex	Average	Average	Easy
	Integrated Efficiency	Fast	Average	Fast	Slow	Average	Average	Fast
	Interaction Design	Fast	Average	-	Average	Plug-in Required	Fast	-
	Build Widgets	Fast	Slow	Fast	Average	Slow	Slow	-
	Device Testing	Fast	Slow	-	Average	Plug-in Required	Average	Fast
Fidelity	Visual Fidelity	Average	Average	Low	High	High	High	High
	Interactive Fidelity	Average	High	-	High	High	High	Average
Professional Skill Requirement	Product Experience	Required	Required	Required	Required	-	-	Required
	Visual Design	-	-	-	Required	Required	Required	Required
	Programming Knowledge	-	Basic Knowledge	-	-	Basic Knowledge	-	-
Sharing		Average	Great	Average	Great	-	-	Great

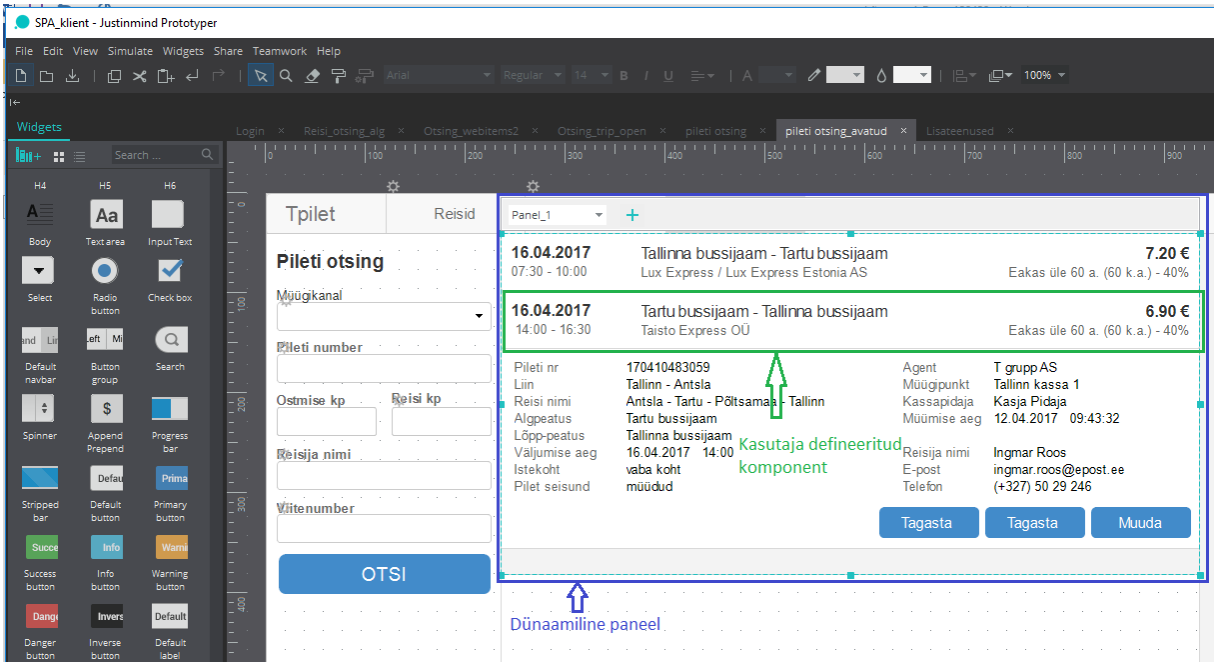
Joonis 4 Prototüüpimise tarkvarade võrdlus [7]

Katsetatud toodetest välistas autor üsna kiiresti prototüüpimisvahendid InVision ja Balsamic Mockups. Esimene nendest langes kõrvale, kuna eeldab kasutajaliidese vaadete eelnevat disainimist mõnes graafikatöötluse programmis ning võimaldab üksnes defineerida pildifailidel interaktiivsed alad, millele klõpsamisel suunatakse kasutaja järgmisele ekraanivaatele. Nii on võimalik küll luua realistlikuna näiva interaktsiooniga kasutajaliideseid, kuid see eeldab iga ekraanivormil toimuva liigutuse jaoks eraldi ekraanivaate kujundamist mõnes graafikatöötluse programmis. Selline lähenemine on sobilik kiiresti graafilist disaini luua suutvatele prototüüpijatele, kelle hulka autor ei kuulu. Balsamic Mockups jäi kõrvale, kuna võimaldab luua kasutajaliidese graafikat üksnes primitiivsel tasemel. Nimetud toote eripäraks ongi käsitsi visandatud sketšidele sarnanev graafiline stiil, mis teatud juhtudel võib isegi kasulikuks osutuda – näiteks aidata

[1] Võrdlus on koostatud 2016. aastal

vältida IT tehniliste teadmisteta kliendi väärarusaama, et prototüübi näol on enamik arendustööd juba tehtud.

Kaks järelejäänud tarkvara olid mõlemad ulatusliku funktsionaalsusega tooted, mis katsid hästi autori vajadusi. Lõplik valik langes tarkvarale JustInMind Prototyper, mille puhul oli kasutaja poolt defineeritavate komponentide loomine käepärasem ja kiirem. Oluliseks eeliseks JustInMind Prototyperi puhul oli võimalus kirjeldada ühe ekraanivaate siseselt dünaamilisi paneele. Tänu sellele ei olnud vaja luua iga interaktsiooni illustreerimiseks eraldi ekraanivormi vaid sai kasutada komponendipõhist lähenemist – ühte kasutajaliidese paneeli puudutav käitumine oli võimalik kirjeldada selle paneeli siseselt.

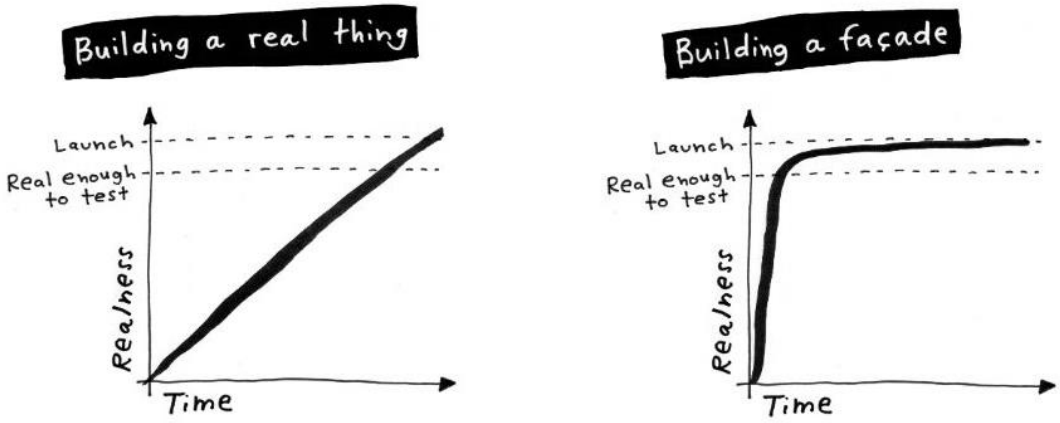


Joonis 5 JustInMind Prototyperi kasutajaliidese ekraanivaade

Tuleb siiski rõhutada, et käesoleva töö eesmärkidest lähtuvat prototüüpimistarkvara valikut ei saa käsitleda universaalsena. Puudusena võib välja tuua funktsionaalselt võimeka toote vilunud kasutamiseks vajaminevat üpris pikka õppimiskõverat. Samuti on detailse graafikaga interaktiivsete ekraanivaadete loomine oluliselt ajakulukam kui baaselementidest koosnevate *wireframede* ehitamine. Sõltuvalt konkreetsest arendusprojektist ning kliendi ootuste määramatuse tasemest tuleb igakordselt kaalutleda, kui palju aega on mõistlik kulutada prototüübi loomisele.

IT visionär Eric Ries on prototüübi loomiseks kulutatava optimaalse aja kohta öelnud: „*Protüüpida ei tasu midagi sellist, mida on hiljem kahju ära visata. Tuleb arvestada*

võimalusega, et pakutav lahendus ei pruugi toimida. Seega ei tohiks anda järele kiusatusele kulutada ideaalse prototüübi valmistamiseks paar lisapäeva või –nädalat. Projekti investeeritud ajal on kahanev lisandväärtus, kuid selle käigus kiindub prototüübi looja aina enam enda tehtavasse lahendusse, mis lõppkokkuvõttes ei pruugi toimida [8].“ Lõpliku toote ja kasutajaliidese prototüübi (näilise fassaadi) kliendipoolse testimiseks kõlbliku versiooni loomiseks vajaminevat ajakasutust illustreerib Ries alljärgnevalt:



Joonis 6 Lõpliku toote ja prototüübi arendamise optimaalne aeg [8]

6. Kasutatavuse arendamine

6.1. Tarkvara kasutatavuse põhimõtted

Tarkvara kasutatavus (*Usability*), kasutajakogemus (*UX*) ja kasutajakogemuse disain (*UXD*) on atraktiivsete lühenditega kaunistatud terminid, millega tarkvara, eriti aga veebipõhiste tarkvaratoodete arendamisega seotud inimesed sageli kokku puutuvad. Erinevate terminite tähendus on praktikute sõnavaras tihipeale kattuva tähendusega. Valdkonnapõhisest erialakirjandusest nähtub, et eeltoodud terminite kohta leidub suur hulk erinevaid definitsioone, mistõttu on nende praktiline kasutamine kattuvas tähenduses üpris ootuspärane. Erinevatest definitsioonidest ühisosa otsides võib siiski väita, et kasutajakogemus on süsteemi (laiemas tähenduses ka teenuse või toote) kasutatavuse põhjal tekkiv emotsioon ja kasutatavus on süsteemi toimimist iseloomustavate omaduste kogum, mis seda emotsiooni kujundab. Vältimaks terminite vahel laveerimist käsitletakse käesolevas töös järgnevalt üksnes loodava tarkvara kasutatavust (mille õnnestunud realiseerimisega kaasneb eeldatavasti ka positiivne emotsioon).

„Hea kasutatavus tähendab, et mingit süsteemi, veebilehte, füüsilist toodet (näiteks teekannu või parkimiskella) või teenust on lihtne kasutada ning raske unustada“ [9].

Kasutatavus algab mõtteviisist, et toode peab olema disainitud selliselt, et selle kasutamine pakub kasutajale maksimaalset rahulolu. Hea kasutatavuse loomeprotsess algab sellest, et tuleb tuvastada, kes on süsteemi kasutajad, aru saada nende vajadustest ja eesmärkidest valida õiged viisid nende vajaduste rahuldamiseks [10].

Paraku ei ole kasutajad ühtemoodi käituvad ja mõtlevad masinad, mistõttu ei ole võimalik kõikide kasutajate jaoks ideaalset kasutatavust tagada. Teine asjaolu, mis ideaalse kasutatavuse väljatöötamist takistab on asjaolu, et kasutatavuse lahendused käivad käsikäes tehnoloogia arengu ja *de facto* standardite kujunemisega. Toome siinkohal kaks näidet:

- Kõigest kümmekond aastat laiatarbe kasutajatele kättesaadavad mitmepuute *(multitouch)* ekraanid pakuvad võimalust luua sellist kasutatavust, mis varasemalt

ei olnud sobiva riistvara puudumise tõttu võimalik. 2012. aastal, mil *multitouch* ekraanide hinnatase oli tänasest kordi kõrgem juurutas TG Tallinna bussijaamas piletimüügiautomaadid, millel on ühepuute ekraanid ja väljumiste nimekirja kerimine võimalik ainult ekraani paremas servas asuva kerimisriba abil. Omas ajas suurepärast kasutatavust pakkunud piletiautomaadid on tänaseks aegunud, sest enamik kasutajaid on harjunud ekraanil kuvatavat kerima suvalisest ekraani punktist kinni haaramise teel ja olemasolev lahendus ajab nad segadusse.

- Sajandivahetuse paiku kasutusel olnud veebipõhist otsingut sisaldavates süsteemides võis pidada suurepäraseks kasutatavuseks seda, kui otsinguväljale sai sisestada mittetäielikke otsingusõnasid ning süsteem kasutas otsinguid realiseerivates päringutes *wildcarde*[1] sisestatud väärtust sisaldavate tulemite leidmiseks. 2004. aastal juurutas Google enda otsingumootoris asünkroonse XMLHTTP Javascripti päringu, mida tänapäeval tunneme lühendi AJAX *(Asynchronous JavaScript and XML)* nime all [11]. Ühtäkki läks see juba 1998. aastal loodud tarkvaratehnoloogia massidesse ja kujunes hea kasutatavuse *de facto* standardiks.

Hea kasutatavuse ajas muutuvuse ja subjektiivsuse faktorile vaatamata on võimalik luua kasutatavust, mida enamik kasutajaid hindab vähemalt antud ajaperioodis suurepäraseks. Selle tulemuseni jõudmiseks tuleb tuvastada need omadused, mis on süsteemi kasutajate ja/või äriprotsesside seisukohalt olulised ning leida saadaoleva tehnoloogia raames parimad praktikad, mis taotletavat omadust parimal moel toetavad.

Whitney Quuensbery on kasutatavuse seisukohast oluliste omaduste lahtimõtestamiseks loonud 5E mudeli, mis koosneb alljärgnevast [10]:

- **Effective**[2] (Tulemuslik) –võimaldab täita täpselt ja tulemuslikult süsteemi kasutaja eesmärki.
- **Efficient** (Tõhus) – võimaldab tegevusi teostada kiirelt ja efektiivselt.
- **Engaging** (Tähelepanu haarav) – tõmbab kasutaja tähelepanu protsessi seisukohalt olulisele. Aitab kasutajal töövoogu sisse elada.

[1] Metakaart – tundmatut või suvalist väärtust tähistav märk andmebaasisüsteemidest päringute tegemiseks.
[2] Alges tähenduse ja mudeli autori taotletud turundusliku eesmärgiga sõnademängu säilitamiseks on komponentide nimetused esitatud inglise keeles.

- **Error tolerant** (Tolerantne) – ennetab ja väldib kasutajapoolsete vigade tekkimist. Nende avaldumisel aitab kasutaja tagasi ootuspärasele töövoole.
- **Easy to learn** (Lihtsasti õpitav) – süsteemi funktsionaalsus on kasutajale lihtsasti omandatav.

Autori hinnangul on see suurepärane raamistik kasutatavuse kitsaskohtade otsimiseks. Olemasoleva KMR-i puuduste analüüsimine 5E mudeli raamistikus näitas, et peatükis 3.1 kirjeldatud puudused liigituvad just eeltoodud omaduse vajakajäämiste alla. Näiteks väljatoodud puudus, et müügiprotsessis vajaminevate klahvivajutuste arv ei ole optimaalne liigitub halva tõhususe alla.

Siiski, 5E mudelis sisalduvad omadused ei ole praktikas teineteisest sõltumatud ja seetõttu tuleb hea kasutatavuse välja töötamisel otsida eesmärgipärast tasukaalupunkti nende omaduste vahel.

„Näiteks õpitavus ja tõhusus on üksteisest erinevad ja mõnikord töötavad üksteisele ka vastu. Seetõttu käib nende vahel pidevalt ka väiksemat sorti vägikaikavedu“ [9]. On ülimalt loogiline, et erinevate omaduste vaheline tasakaalupunkt paikneb erinevate süsteemide puhul ka erinevates kohtades. Näiteks ei ole mõistlik taotleda ulatusliku funktsionaalsusega 3D joonestustarkvarale sama lihtsat õpitavust kui kindlustuspakkumuste võrdlemiseks loodud veebikeskkonnale.

Järgmine peatükk keskendub bakalaureusetöö raames loodud KMR-i kasutajaliideses selliste disainimustrite ja komponentide toimeloogika valimisele, mis koostoimes tagavad parima tasakaalu 5E mudelisse kuuluvate omaduste vahel selle konkreetse tarkvara kasutajaid, kasutamise eesmärki ja funktsionaalsuse ulatust arvestades.

6.2. KMR-i kasutatavus

Alljärgnevalt on esitatud mõned konkreetsed näited KMR-i prototüübi väljatöötamisel rakendatud põhimõtetest ja tehnilistest võtetest, mis aitavad kaasa rakenduse hea kasutatavuse tagamisele. Suurepärase kasutatavuse saavutamiseks on võimalik kasutada oluliselt laiemat hulka disainivõtteid. Siinkirjeldatu on autori valik lahendatava ülesande kontekstis tõhusaimatest võtetest mitte universaalne kasutatavuse arendamise tööriistakast.

Järjepidevus

Kasutajaliidese järjepidevus hõlmab komponentide paigutuse, graafilise disaini, interaktsiooni, kuvatava info detailsuse, struktureeritusse ja kõige muu inimkasutaja ja arvuti vahelist suhtlust mõjutava osas läbivalt ühetaolise ning kasutaja jaoks ootuspärase joone hoidmist. Õigupoolest on järjepidevus kasutajaliidese disaini ja toimimisega nii läbivalt seotud, et lihtsam on tuvastada järjepidevuse puudumist kui defineerida kasutajaliidese osi ja omadusi, mille puhul järjepidevuse nõuet tasuks järgida.

Järjepidevuse tagamine on oluline, kuna seeläbi on võimalik lühendada süsteemi kasutaja õppimiskõverat. Järjepidevalt ülesehitatud kasutajaliidese puhul ei pea kasutaja õppima, kuidas iga üksikut operatsiooni teostada, sest ühe tegevuse selgeks õppimise järel on järgmise läbiviimine kogemuse põhjalt tuletatav. Samuti aitab järjepidevus maandada kasutaja ebakindlust süsteemi kasutamisel [12].

Üheks järjepidevuse tagamise mooduseks on töötada ise välja või kasutada mõnda olemasolevat kasutajaliidese elementide stiili defineerivat teeki ning juurutada seeläbi projektis kasutatavate ekraanielementide kujunduse reeglistik. KMR-i puhul otsustas autor kasutada *Twitter Bootstrap* CCS *(Cascading Style Sheets)* teeki. Valiku tegemisel mängis rolli varasem kokkupuude vastava teegiga ning asjaolu, et *Bootstrap* oli käesoleva töö kirjutamise ajal maailma enimkasutatud CSS teek, mida on rakendatud ca 15% veebisaitide puhul [13]. Niivõrd suure populaarsusega teek sisaldab ootuspäraselt ka suure hulga elementide definitsioone, mistõttu tuli järjepidevuse tagamiseks teha nendest mõistlik valik.

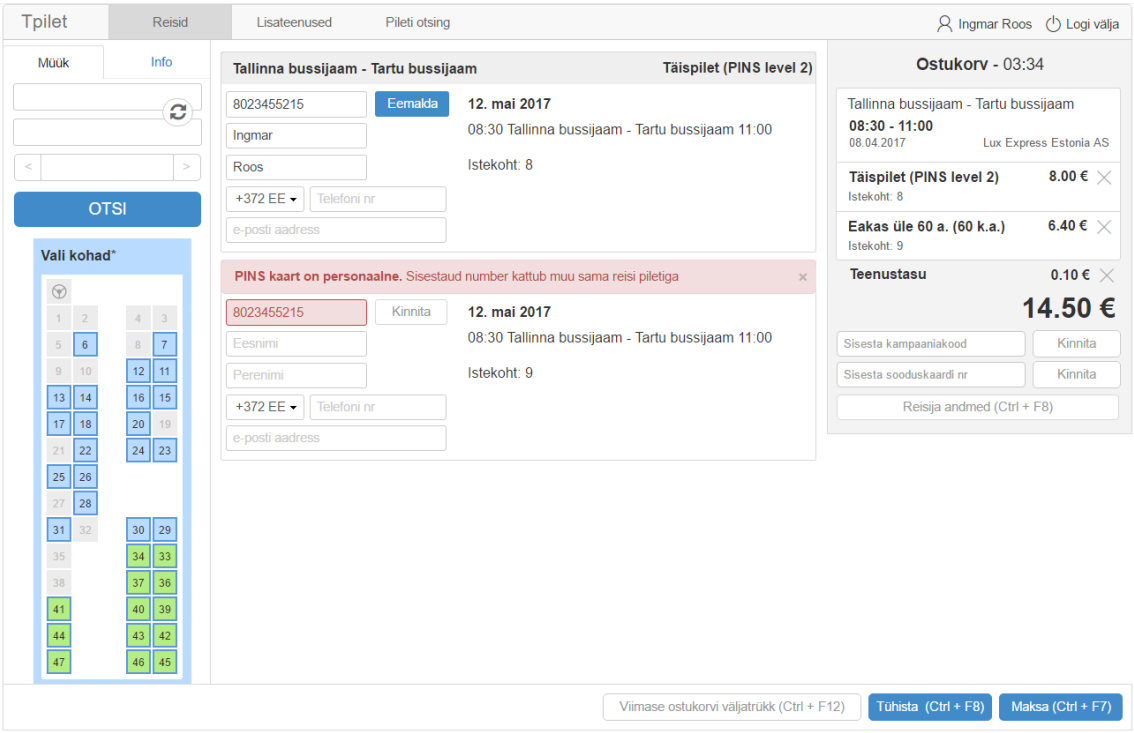
Järjepidevuse tagamiseks ei piisa siiski nuppude ja muude kasutajaliidese elementide ühtse disaini defineerimisest. Järjepidevust tuleks käsitleda teiste kasutatavust parandavate tehnikate ülesena – näiteks, kui ühes kohas on kasutatud sisestusväljade proaktiivset valideerimist, siis tuleks seda teha ka kõikide teiste sisestusväljade puhul.

Proaktiivne valideerimine

Ilmselt on iga veebikeskkonnas registreerimisvormi täitnud arvutikasutaja puutunud kokku groteskse olukorraga, kus arvukate sisestusväljade täitmise ja „Kinnita“ nupule vajutamise järel annab veebileht vea, sest valitud parooli pikkus ei ole piisav. Peale korrigeerimist selgub, et parool peab sisaldama suurtähte ja veidi hiljem, et see peab

sisaldama ka numbrit. Proaktiivse valideerimise eesmärgiks on andmete formaadi ja loogilisuse nõuetele mittevastavuste tuvastamine võimalikult varakult, et nende avaldumine ei näiks süsteemi kasutajale ootuspärast töövoogu häiriva takistusena.

KMR-i puhul on proaktiivne valideerimine realiseeritud seeläbi, et kohustuslikud andmeväljad on arusaadavalt tähistatud ning andmete kinnitamise nupud on kuni sisestatud väärtuste valideerumiseni mitteaktiivsed. Samuti toimub *frontendis* jõustatud ärireeglite piires andmete formaadi ja loogilisuse kontroll ning valideerimisreeglitele mittevastavad väljad tähistatakse veasituatsioonile viitava punase värvi ning vea sisu selgitava teatega kohe, kui kasutaja nendelt edasi liikuda püüab.



Joonis 7 Proaktiivne valideerimine KRM-s reisija andmete sisestamisel

Klaviatuuri kiirkäskude rakendamine

KMR-i puhul on töö efektiivsuse ja sujuva klienditeeninduse tagamiseks oluline, et tarkvara abil oleks võimalik sageli teostatavaid põhiprotsesse läbi viia võimalikult kiiresti. Seetõttu oli üheks eelduseks see, et piletimüügi protsessi peab olema võimalik läbida ainult klaviatuuri kiirkäske kasutades. Klaviatuuri kiirkäsud on realiseeritud ka olemasolevas KMR-s ning süsteemi kasutajatel on tekkinud nende kasutamise vilumus. Paraku ei ole loodava tarkvara puhul võimalik säilitada kõikide operatsioonide väljakutsumiseks samasid klahvikombinatsioone, sest veebibrauseritel on

funktsiooniklahvid reserveeritud brauseri operatsioonide väljakutseks (näit <F5> käivitab enamikus brauserites lehe uuesti laadimise, <F11> täisekraani vaate jne). See kitsendus aitas samas välja töötada universaalsema ja läbivalt ühetaolise kiirklahvide süsteemi, kus sama klahv toimib erinevate tegevuste teostamisel samasuguses kontekstis.

Fookuse juhtimine

Desktop rakendusega samaväärse kasutatavusega üheleherakenduste puhul on oluline tagada, et iga operatsiooni teostamise järel aktiveeritakse kasutusloo kontekstis loogiliselt järgmine element või komponent. Samavõrd tähtis on see, et kasutajale oleks arusaadav, milline element on antud hetkel aktiivne. Eriti oluline on see juhul, kui rakenduses navigeerimine toimub olulises osas klaviatuuri abil, sest sel juhul ei määra fookuses olevat paneeli hiirekursor.

Käesoleva töö raames loodud KMR on üles ehitatud kolmest vertikaalsest paneelist koosneva kasutajaliidesena. Vähendamaks vajaminevate klahvivajutuste või hiireklikkide arvu, toimub paneelide vahelise fookuse juhtimine automaatselt kasutusloo kontekstis kõige loogilisemat tegevuste järjekorda eeldades. Näiteks reisi otsingul liigub peale sihtkoha sisestamist ja <enter> klahvi vajutamist fookus automaatselt otsingutulemuste paneelile, millel aktiveeritakse esimene reis. Nii on kasutajale lihtsasti tuvastatav, et ta saab asuda nooleklahve kasutades reiside nimekirjas navigeerima ning ei pea selleks eelnevalt hiirekursoriga vastavale paneelile klõpsama.

Sujuvad siirded

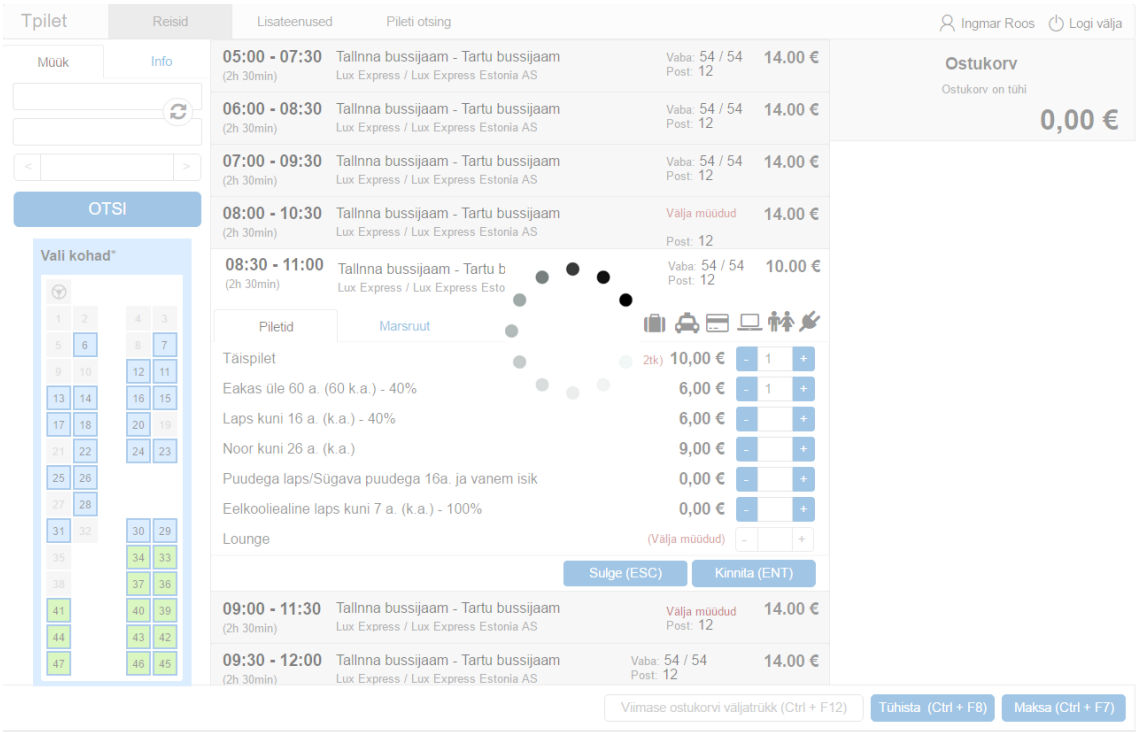
Keerukamate protsesside läbiviimiseks loodud rakendustes on üksteisele järgnevate tegevuste sooritamiseks vaja esitada infot, mida ekraanil korraga kuvada ei ole võimalik või mõistlik. Liiga suure hulga info korraga kuvamine hakkaks kahjustama kasutaja tähelepanu, sest hetkel teostatava tegevuse kontekstis olulist oleks ekraanilt raske leida. Sel põhjusel on vaja kuvatavat infot ja ekraanikomponente muuta ehk tekitada siirdeid.

„Siirded peaksid olema disainitud selliselt, et aitavad kasutajal tajuda enda asukohta kasutajaliideses. Hästi disainitud siirded parandavad kasutatavust läbi selle, et annavad kasutajale kindlustunde rakenduses navigeerimisel – tunde et kasutaja kontrollib olukorda“ [14].

KMR-i kasutajaliidese puhul oli antud teemaga seotud väljakutseks näiteks see, kuidas reiside loendist ühe kindla reisi aktiveerimisel tuua piletihindasid sisaldav ekraanikomponent nähtavale nii, et kasutaja ei kaotaks ülevaadet, kus kohas ta loendi teiste kirjete suhtes asub. Selle probleemi lahendamiseks animeeriti prototüübis ekraanikomponentide ümberpaigutamise selliselt, et konkreetse loendielemendi valimisel nihkub see sujuvalt paneeli ülaosasse ning selle alla ilmub pileti liigi valimiseks ettenähtud ekraanikomponent. Sama toimeloogikat kasutatakse ka muude loendite puhul.

Kasutajaliidese interaktiivsus

Kasutaja tähelepanu haarava ja hoidva kasutajaliidese üheks eelduseks on anda kasutajale tagasisidet, et tema poolt tehtud operatsiooni tulemusena täidetakse mingit käsku. Sarnaselt eelmises punktis kirjeldatule annab see kasutajale kindlustunde, et tema kontrollib süsteemi tööd ning rakendus ei ole seiskunud. Alati ei ole aga kasutaja tegevuse tulemit võimalik kohe kuvada, sest taustal võib toimuda mingi päring, mille vastus on vaja ära oodata. Kui päringud toimuvad esitluskihti uuesti laadimata – näiteks AJAX päringutena, siis võib tekkida olukord, nupuvajutuse tulemusena ei toimu näiliselt mitte midagi. Selliste olukordade vältimiseks kuvatakse loodava KMR-i kasutajaliideses päringute töötlemiseks kuluval ajal ekraanivormide ülekatteid koos laadimisajale viitavate animatsioonide kuvamisega.



Joonis 8 Laadimisaja interaktiivne esitus kasutajaliideses

6.3. Kasutatavuse testimine

„Tarkvara loovate disainerite ja analüütikute suurimaks väljakutseks ei ole mitte selle välja selgitamine kuidas tehnoloogia või äriprotsessid toimivad, vaid see kuidas inimesed tegutsevad. See, mida inimesed ütlevad ja see kuidas nad tegutsevad on kaks täiesti erinevat asja ja ainus võimalus neis mõlemas selgusele jõuda on testida. Kasutatavuse testimine on rohkem kui lihtsalt funktsionaalsete nõuete nimekirjas linnukeste tegemine – see on kõige olulisem suunanäitaja disaini puudutavate otsuste tegemisel“ [15] .

„Kasutatavuse testimine võimaldab disaini- ja arendustiimidel saada teada probleemidest enne kui need koodi kirjutatakse. Mida varem probleemid identifitseeritakse, seda vähem kulukamad on parandused“ [16] .

Tarkvara kasutatavuse testimiseks on välja töötatud suur hulk erinevaid meetodeid. Kasutatavuse testimise ja teadusliku uurimisega tegelevate ettevõtete lipulaevaks peetava Nielsen Norman Groupi kodulehel[1] on kirjeldatud 20 erinevat kasutatavuse testimise meetodit. Lähenemisnurkade rohkuse tõttu ei ole ilmselt ühegi projekti puhul võimalik ega ka mõistlik kasutada kõiki meetodeid – seeläbi saavutaks ebamõistlikult suure jõupingutuse ja kuluga üksnes vasturääkivaid tulemusi.

Üks esimesi samme kasutatavuse testimise juures on välja mõelda testiplaan. Plaani eesmärgiks on dokumenteerida, mida hakatakse tegema, kuidas testi läbi viiakse, millist meetrikat kogutakse, mitu inimest testis osaleb ja milliseid stsenaariume kasutatakse.[9] Testimisplaani keskseks osaks on testimise eesmärkide defineerimine.

Käesoleva töö raames läbiviidud kasutatavuse testimise kavandamisel püstitas autor alljärgnevad eesmärgid.

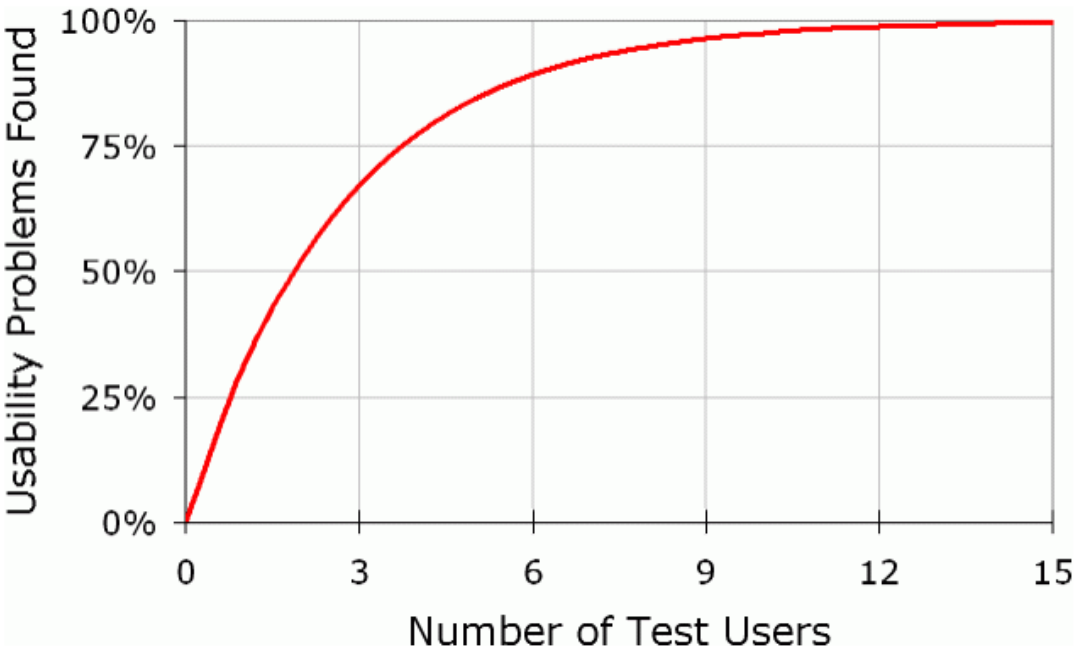
- Tuvastada kõige suuremad kasutatavuse probleemid mis KMR-i prototüübi kasutamisel tekivad.
- Saada kasutajate tagasisidet alternatiivsete lahendusvariantide osas.
- Koguda loodava süsteemi tulevaste kasutajate ideesid kasutatavuse arendamiseks.

Loodava KMR-i tulevasteks kasutajateks on TG ja tema koostööpartnerite klienditeenindajad. Töö koostamise seisuga töötas TG hallatavas Tallinna bussijaamas 11

[1] https://www.nngroup.com/articles/which-ux-research-methods/

klienditeenindajat ning Tartu bussijaamas 4 teenindajat. Lisaks nendele hakkavad loodavat rakendust kasutama TG alltöövõtjate ja frantsiisivõtjatena töötajad, keda on kokku ca 25. Seega on tegemist üsna tagasihoidliku kasutajate koguarvuga. Enamik klienditeenindajaid on juba aastaid töötanud praegu kasutusel oleva kassamüügitarkvaraga, mistõttu on kinnistunud harjumuspärased töövõtted. Neil põhjusel võib tekkida küsimus, kas süsteemi kasutajate arv on üldse piisavalt suur ning testimises osalevad isikud piisavalt avatud, et prototüübi põhjal läbiviidava kasutatavuse testimise põhjal saaks objektiivselt kasutatavust hinnata.

Testimises osalevate kasutajate optimaalse arvu kohta on Jacob Nielsen öelnud, et ulatuslikud kasutatavuse testid on ressursi raiskamine. Parimaid tulemusi annavad võimalikult suure arvu väikeste testide läbiviimine kuni viie kasutajaga. Nielseni kohaselt kogutakse juba esimeselt testis osalevalt kasutajalt kolmandik vajaminevast informatsioonist. Viie kasutajaga testimise korral on võimalik koguda ca 85% informatsiooni süsteemi kasutatavuse probleemide kohta. Kuna testimises osalevate kasutajate poolt välja toodavad probleemid kattuvad, siis iga uue testis osaleva kasutaja käest saadava uue info hulk väheneb. [17] Jacob Nielseni teooriale tuginedes otsustas autor viia kasutatavuse testimise läbi viie TG klienditeenindajaga.



Joonis 9 Avastatud probleemide ja testimises osalevate kasutajate arvu sõltuvus [17]

Teine probleem, mida autor kaalus oli see, et kas olemasoleva süsteemiga harjunud kasutajad on üldse avatud harjumuspärast tööprotsessi muutvateks uuendusteks. Tuli arvestada võimalusega, et vaatamata olemasoleva süsteemi puudustele ning töövoo ebaefektiivsusele, on praeguse tarkvara kasutusloogika nende mõttemallides niivõrd juurdunud, et sellest kõrvalekalduv uus lahendus tundub seetõttu halvem. Samuti võivad testis osalejad tunda ennast ohustatuna, sest pelgavad, et testitava prototüübiga „hakkama saamise" põhjal hinnatakse ka nende kompetentsust – eriti juhul, kui testimist läbiviiv isik on testis osaleja vahetu juht. Selle ohu maandamiseks koostas autor testimise kaaskirja võttes aluseks Steve Krugi poolt 2010. aastal koostatud näidismaterjali, milles on toodud testis osaleja pinge maandamiseks järgmine põhimõte: „*Esimene asi, mida ma tahan rõhutada on see, et me testime uut tarkvara mitte sind. Sinul ei ole võimalik testimise käigus mitte midagi valesti teha"* [18]. Testimise kaaskirjas selgitas autor osalejatele ka uue tarkvara loomise eesmärke ning kinnitas, et kasutajate tagasisidet võetakse arvesse uue töövahendi loomisel. Testimise kaaskiri on toodud bakalaureusetöö lisas 1.

5.5.1. „Mõtle valjusti" meetodil testimine

Kasutatavuse testimine viidi läbi kahes osas. Esimeses osas kombineeris autor „mõtle valjusti"[1] meetodil tüüpiliste ülesannete läbimist A/B testimisega. See tähendab, et kasutajatel paluti teostada sama ülesannet kahel korral, kusjuures kasutajaliidese prototüüp toimis nendel katsetel mõnevõrra erinevalt. Selle lähenemise eesmärgiks oli tuvastada kumb lahendusvariant on kasutajate jaoks käepärasem ning saada kasutajatelt selle kohta selgitavat tagasisidet. Mõlema sooituse ajal tehti testis osaleja kommentaaride põhjal märkusi ning fikseeriti kohad, kus testis osaleja sattus segadusse.

Esimese osa käigus etteantud ülesanded olid sõnastatud kliendi vajadust kirjeldavate stsenaariumite mitte käskivas vormis ülesandepüstitusena. *„Stsenaariumina püstitatud ülesanded on üldjuhul eelistatud sest need aitavad testis osalejatel unustada, et nad sooritavad testi ning vähendavad sellega seonduvat pinget. Tähelepanu tuleks pöörata olukorrale, et etteantav stsenaarium oleks võimalikult realistlik"* [19].

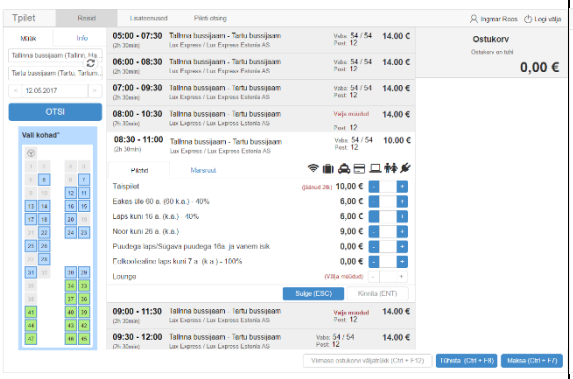
[1] Ingl. Think aloud on kasutatavuse testimise meetod, kus kasutajal palutakse toote või prototüübi kasutamisel kõva häälega kommenteerida, mida ta mõtleb.

Esimese ülesande stsenaariumi kirjeldus: *Klient soovib osta endale ja oma eelkooliealisele lapsele homme kell 14:00 väljuvale Tallinn – Tartu reisile pileteid. Klient soovib istuda bussis enda lapsega kõrvuti asetsevatel istmetel võimalusel bussi eesosas. Klient esitab ostmisel soodustuse saamiseks PINS kaardi numbriga 82452341 ning maksab sularahas 20€ rahatähega.*

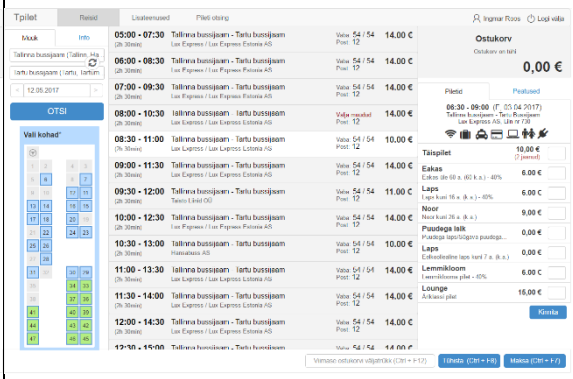
Teise ülesande stsenaariumi kirjeldus: *Klient helistab kassasse ja selgitab, et tal on homme kell 9:00 väljuvale Tallinn – Pärnu reisile pilet ostetud, kuid ta tuleb bussi peale Vana-Pääsküla peatusest Tallinnas. Kliendile bussis koha tagamiseks on vaja vormistada pileti muudatus ilma muutmise teenustasu kohaldamata. Kliendi nimi on Paul Puuraid.*

Klassikalise A/B testi korral suunatakse süsteemi kasutajad juhuslikkuse alusel kahe või enama erineva disainiga versiooni kasutama ning mõõdetakse kumma versiooni korral käitusid kasutajad rohkem eesmärgipärasel viisil. See on levinud tehnika näiteks e-kaupluste ja suure kasutajaskonnaga veebilehtede puhul. Üldjuhul rakendatakse selle meetodi korral suurt hulka kasutajaid, et nende käitumist oleks võimalik mõõta piisaval usaldusväärsuse tasemel. KMR-i testimisel ei olnud eelmainitud põhjusel võimalik kaasata A/B testimisse meetodi tavapärasele rakendamisele omast kasutajate arvu. Seetõttu rakendas autor A/B testimist veidi modifitseeritud kujul − kasutajad läbisid mõlemad stsenaariumid kaks korda, kusjuures esimesel ja teisel korral oli kasutajaliidese ülesehitus mõnevõrra erinev. Autor kui testija täheldas mõlema katse juures üles, milliste kohade juures tekkis kasutajatel segadus ning millist tagasisidet nad „mõtle valjusti“ tehnikaga andsid.

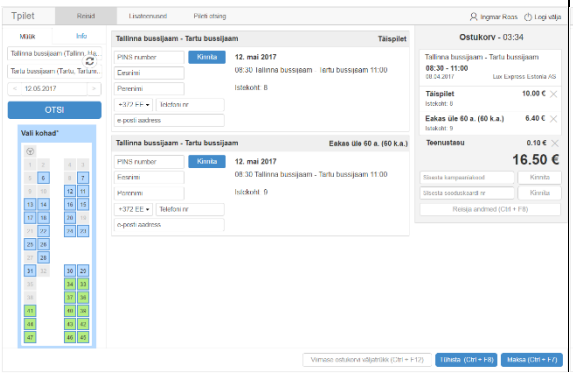
Variant A. Pileti lisamine



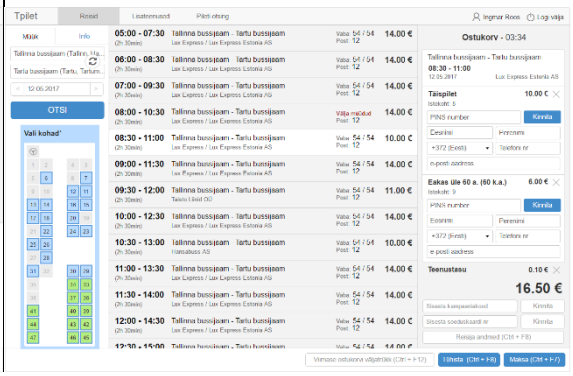
Variant B. Pileti lisamine



Variant A. Reisija andmete sisestamine



Variant B. Reisija andmete sisestamine



Joonis 10 Kasutajaliidese variandid A/B testimisel

5.5.2. Intervjuud

Mitmed kasutatavuse eksperdid rõhutavad, et see mida süsteemide kasutajad ütlevad ja see kuidas nad tegutsevad on kaks täiesti erinevat asja. Jacob Nielsen on toonud välja kaks olulisemat põhjust, miks intervjuud ei ole universaalselt rakendatavad kasutatavuse arendamisel:

- Inimeste mälu on lünklik. Nad ei suuda tagantjärele detailselt meenutada, kuidas nad süsteemi kasutasid ja seetõttu kipuvad nad ratsionaalsema tagasiside andmiseks ning meelest läinud tühikute täitmiseks asju välja mõtlema, et nende jutt kõlaks loogilisemana.
- Inimesed on pragmaatilised ja konkreetsed. Puhtalt süsteemi kirjelduse põhjal ei teki neil mingit ettekujutust kuidas nad kirjeldatud süsteemi kasutaksid. [20]

Ilmselt toimib viimati nimetatu ka vastupidi – enamik kasutajaid ei ole analüütikud ega disainerid ning ei oska ilma visuaalset konteksti omamata kirjeldada süsteemi, mida neil oleks mugav kasutada. Sellest võib järeldada, et intervjuusid ei ole mõtet läbi viia sellises

stiilis, kus kasutajatele kirjeldatakse, mida ja kuidas loodava süsteemiga teha saab ja selle põhjalt paluda hinnangut, kas see oleks sobiv lahendus.

Neid intervjuudele omaseid probleeme õnnestus autori hinnangul KMR-i testimisel edukalt vältida. Kuivõrd testis osalejatele tutvustati täisfunktsionaalset prototüüpi, mille iga nupuvajutus toimis tõetruult, siis langes ära oht, et neil ei teki ettekujutust, kuidas nad süsteemi reaalses elus kasutaksid. Testis osalejate kiiret unustamist puudutavat ohtu maandas autor sellega, et intervjuu käigus käidi eelnevalt läbitud teststsenaariumid prototüübil uuesti läbi. Käesoleva töö puhul pidas autor intervjuude läbiviimist eriti oluliseks seetõttu, et KMR-i tulevased kasutajad tunnetaksid, et neid on arendusprotsessi kaasatud ja oleksid seeläbi vastuvõtlikumad uue tarkvara juurutamisele.

Intervjuud viidi läbi ca 30 minutiliste sessioonidena Tallinnas ja Tartus (Tallinnas 3 kasutajat, Tartus 2 kasutajat). Testimine viidi nii Tallinnas kui ka Tartus läbi testis osaleja loomulikus töökeskkonnas kasutades sama lauaarvutit, mida testis osalejad igapäevases töös kasutavad.

Kõikidele testis osalejatele esitati kolm standardküsimust ning ülejäänud osas arvestas autor küsimuste tõusetumisega intervjuule eelnevalt sooritatud ülesannete baasilt. Samuti paluti prototüübi põhjal esitada ettepanekuid selle kohta, mida tuleks muuta või täiendada. Standardküsimused olid järgmised:

- Nimeta kolm praeguses tarkvaras esinevat probleemi, mis tuleks uues tarkvaras lahendada.
- Kas midagi tundus prototüübi kasutamisel ebaloogilist?
- Kummal sooritusel tundus süsteemi kasutatavus parem ja miks?

Intervjueeritavate hinnangute ja muudatusettepanekute kogumisel arvestas autor sellise nähtusega, mida Jacob Nielsen on nimetanud „päringu mõjuks“ (*query effect*). See nähtus väljendub selles, et kui inimestelt seda küsida, siis suudavad nad kujundada seisukoha peaaegu kõige osas. Seetõttu võib saada kasutajatelt põhjalikke kommentaare ka asjade kohta, mis nende jaoks tegelikult absoluutselt olulised ei ole ja mille üle nad testitavat süsteemi iseseisvalt kasutades ei mõtlekski. Seetõttu on ohtlik teha süsteemis muudatusi lihtsalt selle põhjal, et „kasutajatele see ei meeldinud“ või „kasutajad soovisid seda“. [20]

„Päringu mõjuga“ arvestamine ei tähenda, et kõik intervjuudel esitatud ettepanekud oleks kõrvale heidetud, vaid seda, et intervjueeritavatel paluti esitatud ettepanekutega koos hinnata 5 palli skaalal kuivõrd oluline on selle realiseerimine nende igapäevase tööprotsessi efektiivsuse seisukohalt.

5.5.3. Testimise tulemused

„Mõtle valjusti“ meetodi ja A/B testimise tulemusena tehtud olulisemad järeldused olid:

- Esimese stsenaariumi kohase ülesande täitmisel tegutsesid testis osalejad sujuvamalt kasutajaliidese variant A puhul, kus piletitoote valik ja reisija andmete sisestamine toimus kasutajaliidese keskmises paneelis. Seda varianti kinnitasid kasutajad eelistavat ka testimise esimesele osale järgevate intervjuude käigus.
- Kasutajad ei suutnud prototüübi põhjal hoomata, et reisi otsingu tulemusena kuvatavates loendites on võimalik liikuda klaviatuuri nooleklahve kasutades. Hiirega navigeerides tekkis kahel kasutajal tõrge piletitoodete valimiseks ettenähtud ekraanikomponendi aktiveerimisel.
- Teisele stsenaariumile vastava ülesande täitmisel tegutsesid testis osalejad sujuvamalt kasutajaliidese variandi A puhul, kus pileti muutmine toimub põhiekraanivaate kohal avatavas modaalaknas[1].

Testimise tulemuste põhjal otsustas autor kasutada kasutajaliidese prototüübi edasisel arendamisel neid variante, mille kasutamisel testis osalenutel vähem tõrkeid tekkis ning mida nad intervjuude käigus ka eelistatuks nimetasid. Teisele väljatoodud probleemile efektiivset lahendust ei õnnestunud leida ning selleks puudub ka vajadus, sest nooleklahvidega nimekirjades navigeerimine on KMR-i prototüübis kõikide loendite puhul ühetaoliselt lahendatud. Selle alusel võib eeldada, et kui kasutajad omandavad teadmise, et nooleklahvidega on võimalik navigeerida ja <enter> klahvi valikuga käivitada aktiivse elemendiga seotud järgmine operatsioon, siis oskavad nad edaspidi klaviatuuri abil navigeerida kõikide KMR-i kasutusjuhtude lõikes. Viimati nimetatud probleemi avaldumine oli autori jaoks üllatuslik, kuna klaviatuuriga navigeerimine on realiseeritud ka olemasoleva KMR-i puhul. Oletuslikult on põhjus selles, et paljudes

[1] Ingl modal, Ekraanikomponent, mis ilmub vaate kohale ja muudab tähelepanu tõmbamiseks alloleva vaate tuhmiks või tumedaks

veebipõhistes süsteemides ei ole klaviatuuri nooleklahvide abil menüüdes navigeerimine võimalik ning kasutajad laiendasid sama loodava KMR-i prototüübile.

Intervjuude käigus kogutud andmete põhjal tuvastas autor järgmist:

- Kasutajate poolt välja toodud olemasoleva oleva KMR-i puudused olid loodava KMR-i lähteülesandes juba kaardistatud ning nende lahendamisega prototüübis arvestatud.
- Neli kasutajat viiest eelistas esimese teststsenaariumi läbimise põhjal kasutajaliidese varianti A.
- Kolm kasutajat viiest eelistas teise teststsenaariumi läbimise põhjal kasutajaliidese varianti A.
- Kasutajad eelistasid oma hinnangutes mõlema teststsenaariumi puhul kasutajaliidese varianti, mille puhul oli ka vaatluspõhiselt nende töövoog sujuvam.
- Klaviatuuri kiirkäskude toimimist kogu protsessi ulatuses pidasid kasutajad oluliseks üksnes pileti müümise põhiprotsessi puhul. Muude tegevuste puhul hinnati hiire kasutamist töö efektiivsust vähe mõjutavaks.

Intervjueeritavate poolt väljatoodud muudatus- ja täiendusettepanekute põhjal viidi prototüübis sisse alljärgnevad täiendused või kirjeldati komponentide toimimist ettepaneku realiseerimiseks arendusetapis.

- Piletite ja lisateenuste koguse määramiseks ettenähtud sisestusväljadele lisati miinus- ja plussklahv, mille abil saab kogust määrata ka hiirega.
- Spetsifitseeriti nõue, mille kohaselt peab paarisarvu piletite ostmise korral rakendus võimaluse korral määrama neile piletitele automaatselt kõrvuti asetsevad istekohad.
- Piletiliikide loendisse lisati info hetkel kehtiva hinnaklassi järelejäänud piletite arvu kohta.
- Piletiliikide loendite järjekorda muudeti selliselt, et väljamüüdud piletiliigid esitatakse nimekirja lõpus.

7. Arendusvahendid

Uue KMR-i kavandamisel kerkis üles küsimus, kas uus rakendus tuleks arendada *desktop* rakendusena või üheleherakendusena ehk lihtsamalt öeldes veebirakendusena. Olemasoleva KMR-i näol on tegemist *desktop* rakendusega. Selline lähenemine otsustati 2010. aastal oma aja ning sel ajal levinud tarkvaratehnika arengutaseme kontekstis. Täpsemalt on vastavaid kaalutlusi selgitatud peatükis 3. Kuigi mõningad nimetatud kaalutlustest on jätkuvalt relevantsed, osutus TG muutunud äriperspektiivide, ettevõtte muu tarkvaraportfelli ja veebirakenduste tarkvaratehnoloogia kiire arengu taustal mõistlikumaks luua uus KMR veebipõhise üheleherakendusena. Mõningad olulisemad põhjused, mis seda valikut mõjutasid olid järgnevad:

- TG ja tema tütarettevõtte T solutions OÜ loodud spetsiaaltarkvarade portfelli ei kuulu teisi *desktop* rakendustena säilinud komponente. Arendusmeeskonna kompetentside hoidmise ja tarkvara hooldamise kulu arvestades ei ole mõistlik *desktop* rakenduse kui erilahenduse hoidmine.
- Veebirakenduste puhul on oluliselt lihtsam läbi viia tarkvara versiooniuuendusi. Kui desktop rakenduse puhul eeldab versiooniuuendus uue versiooni paigaldamist kõikidesse rakendust kasutavatesse arvutitesse, siis veebirakenduse puhul piisab, kui uue versiooni paigaldus[1] teostatakse serveris ning selle tulemusena toimub versiooniuuendus lõppkasutaja vaates märkamatult. [21]
- Veebirakendus toimib sisuliselt kõikidel platvormidel ning operatsioonisüsteemidel, millel on võimalik käitada veebibrauserit. Kehtib põhimõte „kirjuta koodi üks kord, käivita kus iganes“[2]. See põhimõte osutus oluliseks arvestades ettevõtte äriperspektiive pakkuda loodavat KMR-i T solutions piletimüügiplatvormi ühe komponendina välisturgudel. [21]
- Veebirakenduste puhul on lihtsam rakenduse kasutajaliidese disaini ümber kohandada konkreetse kliendi identiteeti ja soove arvesse võttes. [21]

[1] Ingl. *deploy*

[2] Ingl. *„Write once, run anywhere“.* Sun Microsystemsi poolt kasutusele võetud hüüdlause, millega tutvustati Java keele ja Java keeles kirjutatud multiplatvormsete rakenduste eeliseid [56].

Olulisimaks mõjuriks, mis sundis ümber vaatama ca seitse aastat tagasi langetatud otsused oli üheleherakenduste loomist lihtsustavate Javascripti arendusraamistike kiire areng. See on muutnud võimalikuks luua veebirakendusi mille kasutamise mugavus ning kasutajaliidese interaktiivsus on praktiliselt samaväärne *desktop* rakendustega. Esimene olulisem läbimurre selles valdkonnas toimus küll juba 2009. aastal, mil välja tuli multiplatvormne Javascripti käitussüsteem Node.js. [22]. Samas 2010. aastal, kui kavandati praegu kasutusel oleva KMR-i arendamist, oli tegemist veel üpris vähelevinud tehnoloogiaga. Seetõttu ei kaalutud toona tõsisemalt KMR-i veebirakendusena arendamise võimalust.

Viimase 7 aasta jooksul on Javascripti arendusraamistikud teinud läbi tormilise arengu ning arendajad on pigem hädas liiga suure arvu raamistike hulgast sobivaima ja elujõulisima valimisega, kui arendusvahendite puudusega. Peamine argument, mis räägib veebipõhise tarkvara loomisel mõne arendusraamistiku kasutamise kasuks on programmeerimise efektiivsus. *„Funktsionaalust, mille nullist arendamiseks kuluks tunde ning sadu koodiridasid on arendusraamitikus defineeritud funktsioone kasutades võimalik luua mõne minutiga“* [23]. Nimetatud efektiivsuse saavutamiseks on populaarsemate Javascripti arendusraamistike tööriistakastis vahendid, mis aitavad arendajal lihtsa vaevaga lahendada enamikku alljärgnevast [24]:

- Andmeobjektide ja ekraanielementide vaheline andmesidumine (*data binding);*
- Kindla arendusmustri (MVC *(Model-View-Controller)* või mõni selle analoog) järgimisest saavutatud loogiliselt struktureeritud lähtekood;
- Komponentide loomine ja taaskasutamine;
- Marsruutimine *(routing);*
- Sisestusvormi elementide väärtuste valideerimine;
- Lihtne lõimimine väliste süsteemide andmevahetusteenustega;
- Komponentide ja kasutajaliidese elementide animeerimine;
- HTML DOM *(Document Object Model)* sündmuste automatiseeritud käsitlemine.

Loomulikult on arendusraamistiku kasutamisel ka omad kitsaskohad. Nii nagu terminist eeldada võib, peab arendaja raamistiku kasutamisel tegutsema kindlates raamides. Arendusraamistiku tuumikloogikat ei ole võimalik raamistiku lähtekoodi

manipuleerimata muuta. Seetõttu peab arendaja arendusraamistiku kasutamisel aktsepteerima selle piiranguid ja kirjutama koodi nii nagu raamistiku looja on seda ette näinud [23]. Sellest tulenevalt on oluline valida raamistik, mille abil on parimal viisil võimalik täita käimasoleva arendusprojekti nõudeid.

Käesoleva töö koostamisel uuris autor kolme serveripoolse Javascripti arendusraamistiku omadusi ning valis raamistiku, mis erinevaid kaalutlusi arvesse võttes oleks sobivaim KMR-i loomiseks. Nende kolme arendusraamistiku tutvustus ja võrdlus on esitatud järgmises peatükis.

7.1. Arendusraamistiku valik

2017. aasta alguseks oli turule tulnud suur hulk mitmekülgse funktsionaalsuse ja suure arendajate kogukonnaga arendusraamistikke. Tehnoloogiablogi *tecadmin.net* tõi 2017. aasta alguses välja 8 populaarsemat ja elujõulisemat arendusraamistikku, mida *frontend* arendajatel tasuks oma kompetentsi arendamiseks õppida. Märgitute seas on Angular, Aurelia, Ember.js, Knockout.js, Backbone.js, Meteor, Polymer.js ja React [25]. Selles nimistus märgitute näol on tegemist üksnes enamlevinud raamistikega. Muudele allikate tuginedes võis leida andmeid üle 30 Javascripti teegi ja arendusraamistiku kohta, millele ennustati 2017. aastal populaarsuse kasvu [26]. See tõendab, et Javascript arendusraamistike kasutamine veebirakenduste loomiseks on kiiresti kasvav trend. Samas võib eeldada, et niivõrd paljude saadaolevate alternatiividega kiiresti areneval turul on ka mitmeid tooteid, mis paari aasta jooksul konkurentsist välja langevad. Seetõttu on konkreetse projekti realiseerimiseks kasutatavat raamistikku valides oluline arvestada nii raamistiku sobivust kui ka prognoosi arendusraamistiku elujõulisuse osas.

Käesoleva töö eesmärgiks ei ole tarkvaratehnilise süvaanalüüsi läbiviimine kõikide turul saadaolevate alternatiivide võrdlemiseks. Seetõttu on autor piirdunud üksnes kolme arendusraamistiku eeliste ja kitsaskohtade kirjeldamisega neid raamistikke kasutavate praktikute poolt erialakirjanduses väljatoodu põhjal. Samas leidub internetis mitmeid range metoodika alusel Javascipti arendusraamistikke võrdlevaid teadustöid. Eestikeelsena avaldatutest väärib märkimist Erik Räni poolt 2016. aastal avaldatud Tartu Ülikooli bakalaureusetöö „JavaScripti raamistike võrdlus“, milles Räni on võrrelnud kuut raamistikku kindla kriteeriumite kogumi alusel, võttes muuhulgas arvesse selliseid

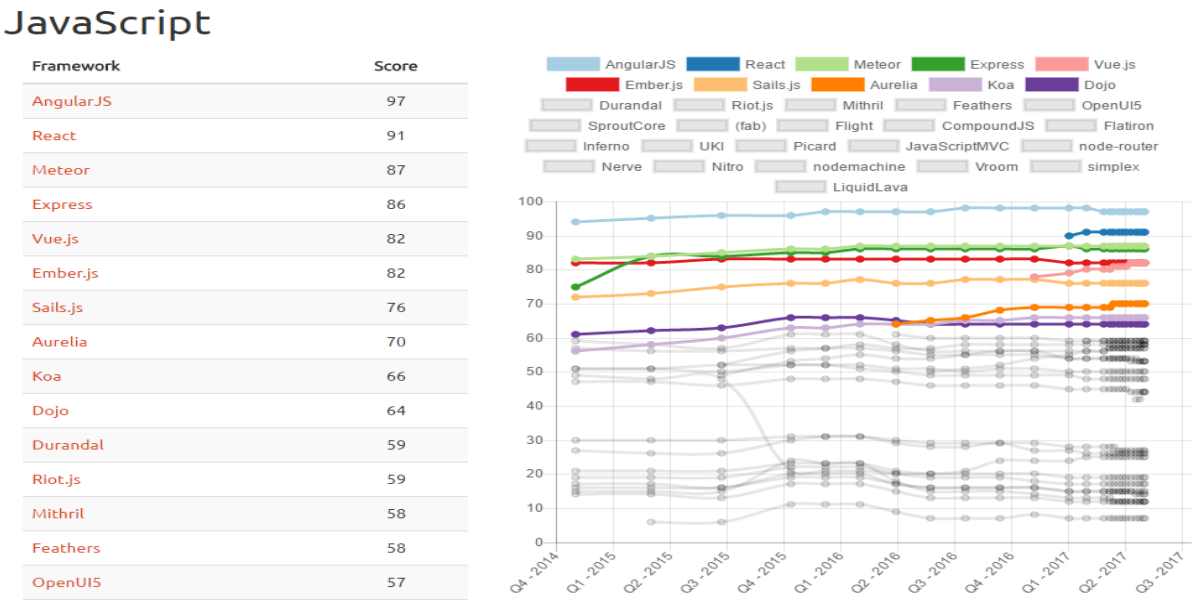
tarkvaratehnilisi näitajaid nagu McCabe'i tsüklomaatiline keerukus[1] ja Halstead'i jõupingutus[2]. Enda poolt püstitatud metoodika põhjal hindab Räni parimaks Javasripti arendusraamistikuks AngularJS-i [27]. Samas nähtub muudest sama uurimisteemat käsitlevatest materjalidest, et parima arendusraamistiku määratlus sõltub hindamisel arvesse võetud parameetritest. Isegi kindla metoodika kasutamisel sõltub tulemus suurel määral sellest, kui hästi sobib ühe või teise arendusraamistiku stiil võrdlevat analüüsi läbiviinud praktiku varasemate kompetentsidega.

Kuna subjektiivse faktori väljataandamine näib autorile võimatu, on käesoleva bakalaureusetöö raames võrreldus arendusraamistikest kaks (Angular ja React) valitud nende populaarsuse alusel. Tehnoloogiate populaarsuse monitoorimisega tegelev veebisait *hotframeworks.com* reastab tehnoloogiaid koondindeksi alusel, mis võtab arvesse koodijagamise keskkonnas GitHub vastava tehnoloogia jälgijate hulka ning StackOverflow[3] keskkonnas tehnoloogia märksõna all püstitatud teemade arvu. Selle meetodi kohaselt oli Javascripti arendusraamistike populaarsuse pingerida 2017. märtsi seisuga järgmine:

[1] Thomas J. McCabe, Sr. poolt 1976 avaldatud kvantitatiivne meetod tarkvara lähtekoodi keerukuse hindamiseks. Hindamismeetod võtab arvesse lineaarselt sõltumatute teekondade arvu läbi programmi lähtekoodi. [Wikipedia2]

[2] Maurice Howard Halstead poolt 1977 aastal avaldatud statistiline meetod, mis aitab hinnata vaimse pingutuse taset, mida läheb tarkvara arendamiseks või ülalpidamiseks vaja programmeerimiskeelte keerukuse mõõtmiseks. [IBM]

[3] StackOverflow on 7,1 miljoni programmeerija osalusel käitatav küsimuste vastuste keskkond. Veebisait: https://stackoverflow.com



Joonis 11 Javascript arendusraamistike populaarsus 2017. a. märtsi seisuga. [28]

Kolmas käesoleva töö raames võrreldav raamistik Aurelia kuulub samuti *hotframeworks.com* pingerea esikümnesse, kuid selle käsitlemise peamiseks põhjuseks on asjaolu, et Aurelia kuulub juba TG tütarettevõtte T solutions OÜ kasutatavate tehnoloogiate virna ning selle baasil on arendatud välja muid ettevõtte tooteportfelli kuuluvaid tooteid. Etteruttavalt võib märkida, et Aurelia kasutamine on KMR-i realiseerimiseks eelistatud valik. Angulari ja Reacti kui kahe turuliidri võrdlemine on läbi viidud selleks, et saada selgust kas seni kasutatud raamistikuga jätkamine on mõistlik.

7.1.1. Angular

AngularJS arendusraamistik lasti Google poolt esmakordselt välja 2010. aastal. AngularJS-i loojate eesmärgiks oli pakkuda täislahendust alates kasutajaliidesest kuni äriloogika realisatsiooni ja testimiseni. Eraldatud on DOM manipuleerimine rakenduse loogikast ning esitluskiht serverirakendusest. 2016. aasta septembris lasti välja Angular (Angular2), mis on Google eraldiseisva meeskonna poolt nullist uuesti kirjutatud raamistik – st sellel ei ole mingit pistmist AngularJS-i ehk Angulari esimese versiooniga. Angulari näol on tegemist hetkel ühe kuumima märksõnaga *frontend* arenduses ja käesoleva bakalaureusetöö kirjutamise seisuga ühe populaarseima *frontend* raamistikuga [29]. Õnnetuseks on Google tekitanud arendusraamistiku nimevalikul korraliku segaduse. Kuigi Angularil on enda eelkäijaga vähe ühisosa, otsustas Google eelkäija populaarsuse tuules jätkamiseks kasutada peaaegu identset nime. Seetõttu on arendajate kogukonnal

üpris keeruline orienteeruda, millised juhendid ja koodinäited käivad AngularJS-i kohta ning millised Angular2 ehk lihtsalt Angulari kohta.

Angulari eeliseks teiste raamistike ees on väga suur arendajate kogukond. Koodijagamise keskkonnas GitHub oli 2017. aasta aprilli seisuga 27 779 Angulari lähtekoodi harutajat[1] ning 55 577 koodiuuenduste jälgijat [30]. Sedavõrd suur arendajate kogukond annab teatava kindluse, et raamistiku kasutamisel tupikusse jooksmisel leiab veebist sarnase teema kohta varasemaid teemapüstitusi koos lahendusega. Angulari suurimaks tugevuseks võib pidada asjaolu, et selle taga seisab suurkorporatsioon Google. Võib eeldada, et sellises mastaabis ettevõtte portfelli kuuluva arendusraamistiku puhul on tagatud läbimõeldud arhitektuur, põhjalik testimine ning jätkusuutlik arendustegevus.

Erialakirjanduses tuuakse Angulari tugevustena korduvalt ka välja hästistruktureeritud koodi ülesehitust, HTMLi ning koodi eraldatust ning suure hulga lisapakettide saadavust.

Angular raamistik on kirjutatud TypeScript[2] keeles ning võimaldab ka arendada rakendusi TypeScripti kasutades. *„Sellega õnnestub programmeerimisel vältida Javascripti kui nõrgasti tüübitud keele omapärast tekkida võivaid vigasid näiteks erinevate andmetüüpide võrdlemisel“* [31].

Märkimisväärseks eeliseks Angulari puhul on ka väga mugav kahesuunaline andmesidumine. See tähendab, et raamistikus on realiseeritud vahendid vaate ja mudeli vaheliseks automaatseks andmesidumiseks. Näiteks juhul, kui kasutaja muudab ekraanivormi sisestusväljal mingit väärtust, siis Angulari vahendid tagavad, et selle tulemusena ei muutu mitte ainult vaate elemendi väärtus vaid ka taustal oleva mudeli väärtus ja vastupidi. See vabastab vajadusest kirjutada eraldi koodi vaate ja mudeli poolel toimuvate muudatuste vastastikuseks ülekandmiseks [32].

Angulari kitsaskohana saab välja tuua selle, et Angular eeldab suures osas koodi algoritmilise poole lahendamist kindlal raamistiku poolt ette deklareeritaval moel. Seetõttu on mitmetel hinnangutel Angular jäik raamistik. Samas ei ole raamistiku poolt eeldatav stiil kuigi lihtsasti hoomatav, mistõttu tuleb juhul, kui mingi

[1] Ingl *forker* – arendaja, kes on loonud koodijagamiskeskkonnas lähtekoodi koopia, millega tal on võimalik vabalt eksperimenteerida ilma, et see mõjutaks algse projekti koodi.
[2] Microsofti poolt Javascripti baasil loodud tugevasti tüübitud programmeerimiskeel, mille lähtekoodi on võimalik kompileerida tavaliseks JavaScripti koodiks [58].

valmisprogrammeeritud koodilõik ei toimi soovitud viisil, üritada sama asja muud moodi lahendada või otsida võimalusi kuidas Angulari enda lähekoodi muuta.

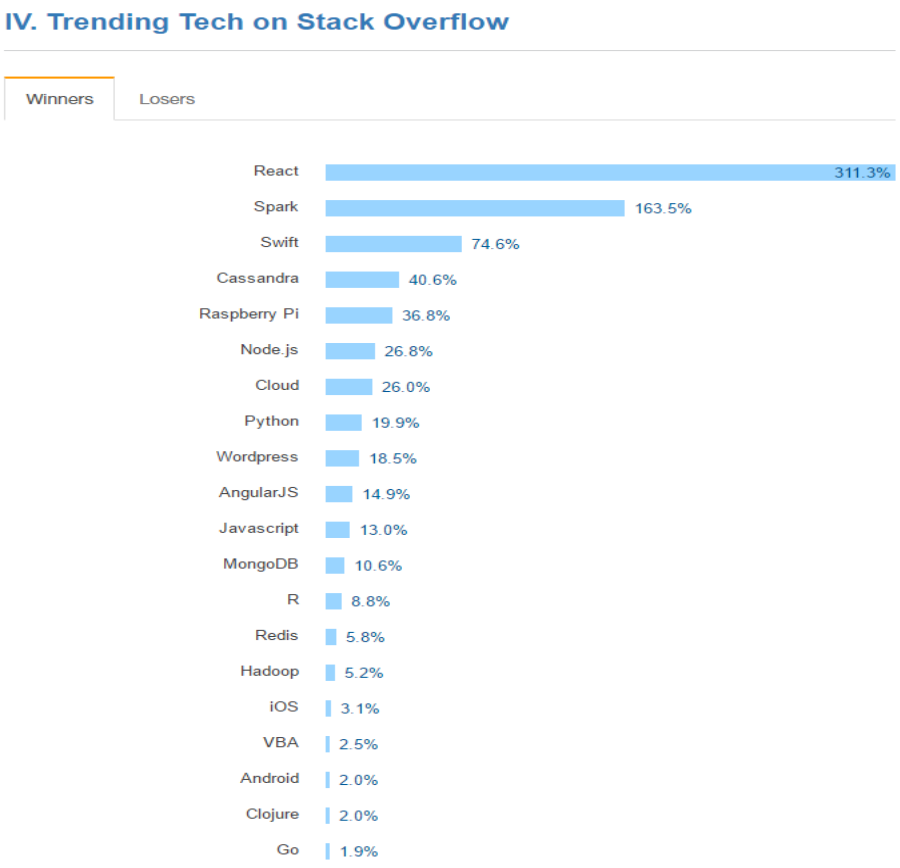
Eelmises lõigus väljatoodud puuduse tõttu hinnatakse Angulari õppimiskõverat teiste populaarsete raamistikega võrreldes suhteliselt pikaks ning mitmete arendajate hinnangul ka järsuks. See tähendab, et näitete põhjal esmaste koodijuppide arendamiseks kompetentsi omandamine näib lihtne, kuid „päris“ rakenduste loomiseks tuleb omandada suurel määral teadmisi raamistiku toimimise kohta [33].

Lisaks eeltoodule on Angulari puhul toodud välja halba jõudlust suuri andmehulki sisaldavate vaadete graafilisel esitamisel brauseris (ehk renderdamisel). Puuduseks võib pidada ka asjaolu, et erinevalt Reactist ei ole Angularis realiseeritud virtuaalset DOM-i [34].

7.1.2. React

React on avatud lähtekoodiga Javascripti teek, mille on loonud Facebooki arendusmeeskond. Teegi loojate väljaütlemiste kohaselt oli Reacti loomise eesmärgiks lahendada küsimus, kuidas luua suuri veebirakendusi, mis käsitlevad ajas muutuvaid andmeid – see on miski millega sotsiaalvõrgustikud nagu Facebook ja Instagram kokku puutuvad. Lisaks mainitutele on Reacti abil enda kasutajaliidesed välja arendanud nimekamate keskkondade hulgas ka näiteks NetFlix, Dropbox ja Airbnb [35].

Reacti ja Angulari vahel käib arendustehnoloogia populaarsuse vaates jõuline vägikaikavedu. Koodijagamise keskkonnas GitHub oli 2017. aasta aprilli seisuga 12 246 Reacti lähtekoodi harutajat ning 66 033 koodiuuenduste jälgijat [30]. StackOverflow keskkonna teemakäsitluste arvu põhjal hinnatuna oli tegemist 2016. aastal kõige kiiremini populaarsust võitnud arendustehnoloogiaga.



Joonis 12 Arendustehnoloogiate populaarsuse kasv 2016 vs 2015 [36]

Erinevalt Angularist ei saa Reacti üksikult võttes nimetada arendusraamistikuks vaid teegiks. Selleks, et üritada tõmmata selgejoonelist piiri nende kahe mõiste vahel toome siinkohal ära Martin Fowleri vastavad definitsioonid:

- *teek on hulk funktsioone, mida saab vajadusel välja kutsuda;*
- *raamistik on abstraktse disaini kehastus, millesse sisestatakse konkreetsetesse kohtadesse raamistiku koodi poolt kasutatavat arendaja loodud koodi* [37].

MVC mudeli vaates tegeleb React peamiselt V-ga ehk vaadetega. Tema algne põhieesmärk oli tagada veebilehe HTML elementide sujuv renderdamine olukorras, kus taustal olevad andmed muutuvad. React hoolitseb kasutajaliidese uuendamise eest nii, et arendaja ei pea selle saavutamiseks realiseerima koodis käsitsi andmesidumist [38].

Samas, kui öeldakse, et mingi veebirakenduse arendamiseks on kasutatud Reacti, siis ei peeta selle all peaaegu kunagi silmas, et kasutatud on ainult algset Reacti teeki vaid tervet virna teeke ja raamistikke. Sellise lähenemise puhul on ilmseks puuduseks asjaolu, et erinevatel praktikutel on erinev nägemus sellest, millest koosneb ideaalne Reacti

arendusvahendite virn. See on samaaegselt Reacti tugevus ja nõrkus. Ühest küljest on rangete raamistike (nagu Angular) probleemiks see, et kui arendajale ei sobi mingi osa raamistikust, siis tuleb kulutada palju energiat „vastuvoolu ujumisele“, et panna raamistik toimima selliselt nagu konkreetse projekti kontekstis vaja. Reactiga sellist probleemi ei ole, sest arendajal on võimalus kasutada koos Reactiga suurt hulka muid arendustehnoloogiaid. Samas on suure hulga erinevate teekide ja raamistike hulgast käimasoleva arendusprojekti jaoks sobivaima valimine ja nende omavaheline integreerimine üpris tülikas [39] .

Reacti suurimaks eeliseks võib erialakirjanduse põhjal pidada esitluskihi kiireks renderdamiseks välja töötatud virtuaalse DOM-i kasutamist. DOM-i manipuleerimine on kaasaegse interaktiivse veebi lahutamatu osa. Õnnetuseks on see ka üks aeglasemalt teostatavaid Javascripti operatsioone. DOM-i uuendamise muudab veelgi ebaefektiivsemaks see, et enamik arendusraamistikke uuendab DOM objekte esitluskihiga seotud andmete muutumisel oluliselt rohkem kui oleks optimaalne. Näiteks juhul, kui 10 elementi sisaldavas loendis üks element muutub, renderdab enamik arendusraamistikke uuesti kogu loendi. Selline tegevus kasutab 10 korda rohkem ressurssi kui oleks optimaalne. Selle probleemi lahendamiseks ongi Reacti puhul välja töötatud virtuaalne DOM, mis on sisuliselt päris DOM-i koopia. Juhul, kui Reacti koodis kirjeldatud elementi muudetakse, siis uuendab React kõiki virtuaalse DOM-i elemente ning võrdleb seda muudatusele eelneva virtuaalse DOM-i hetktõmmisega. Võrdluse tulemusena tuvastab React need elemendid, mis muutusid ja uuendab ainult neid DOM objekte [40].

Virtuaalne DOM on väga efektiivne lahendus suuri ja muutuvaid andmeloendeid sisaldavate veebirakenduste esitluskihi sujuvaks uuendamiseks. Käesoleva bakalaureusetöö raames käsitletava KMR-i puhul ei paku virtuaalne DOM paraku olulist lisandväärtust, kuna käsitletavas rakenduses ei esine pikkasid ja keerukaid graafilisi elemente sisaldavaid loendeid, mille uuendamisel võiks tekkida jõudlusprobleeme.

Reacti eelistena on läbitöötatud erialakirjanduses välja toodud ka seda, et Reacti komponentide kirjeldamiseks kasutatav omapärane, HTML-le sarnanev Javascripti dialekt JSX on väga lihtsasti õpitav ning võimaldab luua taaskasutatavaid komponente, mille koodile otsa vaadates on kompetentsele arendajale lihtsasti hoomatav, kuidas komponent esitluskihis renderdatakse [41].

Reacti puudusena võib lisaks selgelt määratlemata arendusvahendite virnale välja tuua samast probleemist tingitud asjaolu, et arendajate kogukond ei ole välja töötanud parimaid praktikaid Reacti koodi kirjutamiseks. Samuti võib Reacti vahenditega arendamisel üleskerkivate probleemide korral olla raske leida selgeid vastuseid, kuna arendajad kasutavad Reacti koostoimes erinevate teekide ja raamistikega, millede puhul võib sama probleemi lahendus olla erinev.

7.1.3. Aurelia

Aurelia arendusraamistiku loojaks on Rob Eisenberg, kes on oli algselt ka Angulari teise versiooni arendustiimi liige. Eisenbergi loetakse ka populaarsete .NET-i esitluskihi raamistike Caliburn ja CaliburnMicro loojaks [42]. Aurelia loojad on tutvustanud raamistikku hüüdlausetega *„Järgmise generatsiooni kasutajaliideste loomise raamistik“* ja *„Kõige progressiivsem ning arendajasõbralikum frontendi arendusraamistik“* [43].

Jättes kõrvale turunduslikes sõnumites lubatu, väärib Aurelia puhul esiletoomist mitu olulist tugevust. Aurelia on modulaarne raamistik. Erinevalt kindla toimeloogikaga monoliitsetest raamistikest koosneb Aurelia väikestest kindla ülesandega moodulitest, mis on omavahel seotud selgelt defineeritud liideste abil. See võimaldab arendajal kohandada enda projekti jaoks vajaminevat paketti, kasutades ainult neid mooduleid, mida tal vaja on. Samuti võib arendaja vajadusel asendada mõne mooduli enda poolt looduga ning siduda selle teiste moodulitega raamistikus defineeritud liideste abil [44].

Võrreldes Angulariga on sellise lähenemise eeliseks asjaolu, et juhul, kui tekib vajadus raamistikku konkreetse projekti vajadustest lähtuvalt kohandada, siis on muudatuste mõju kogu raamistiku toimimisele lihtsamini prognoositav.

Aurelia eeliseks võib pidada ka asjaolu, et raamistik võimaldab kirjutada lähtekoodi nii Javascripti uusimale standardile vastava ECMAScript[1] 2016 süntaksi järgi, TypeScriptis (sarnaselt Angularile) aga ka Javascripti varasematele standarditele vastavas algses Javascriptis, mida arendajate kogukond nimetab *vanilla javascriptiks* [45]. Võimalus kirjutada komponentide spetsifikatsioone algupärases Javascriptis aitas autoril, kui

[1] Javascripti keele ametlik nimi, mis viitab keele ametlikke standardeid haldava organisatsioonile ECMA [55].

Javascripti uusimaid standardeid ja TypeScripti mitte tundval isikul, lihtsamini hoomata raamistiku poolt eeldavat koodistruktuuri ja toimeloogikat.

Lühike õppimiskõver ja lihtsus ilma funktsionaalsuses järeleandmisi tegemata oli Aurelia loojate üks eesmärke. Raamistiku kodulehel tuuakse ühe eelisena välja: *„Lihtne kuid mitte lihtsakoeline“ (Simple, But Not Simplistic) – Aureliat on lihtne õppida, kuid see on selle vaatamata võimas. Lihtsa ja järjepideva stiiliga disaini tõttu võivad arendajad õppida ära vaid käputäie arendusmustreid ja liideseid, mida on võimalik kasutada lugematul arvul viisidel* [46].

Siinkohal on asjakohane korrata peatükis 6.2 toodud järjepidevuse põhimõtet, mis oli eelnevalt ära märgitud kasutajaliidese disaini kontekstis, kuid mille tähendus on ilmselt laiem. Järjepideva kasutusloogikaga süsteemid on nende kasutajatele lihtsamini õpitavad, kuna võimaldavad süsteemi kohta juba omandatud teadmisi üle kanda järgmistele süsteemi osadele.

Käesoleva töö koostamise käigus läbitöötatud Aurelia koodinäidete baasil hindab autor raamistiku tugevuseks ka REST andmevahetusteenustega lõimimise lihtsust. Raamistiku kodulehe ja arendajate kogukonna postituste seast leidis mitmeid näited päringute autoriseerimise, päringu päiste defineerimise, JSON formaadis andmevoo koostamise ja töötlemise kohta. See on käesoleva projekti kontekstis oluline argument, kuna T solutions piletimüügiplatvormi andmevahetuskiht on ehitatud üles just REST andmevahetusteenustel, millega loodav KMR tuleb lõimida.

Aurelia osas läbitöötatud erialakirjanduses kõige sagedamini mainitud puuduseks on väike arendajate kogukond, mistõttu on spetsiifiliste probleemide korral raske leida koodinäiteid või saada abi foorumitest. Aureliat ei ole käesoleva töö kirjutamise seisuga kasutatud veel ka ühegi „raskekahurväe“ hulka kuuluva veebikeskkonna arendamisel [47]. Kuivõrd Javascripti arendusraamistikke tuleb aina juurde, siis on oht, et suurepärasele kontseptsioonile vaatamata langeb Aurelia vähese populaarsuse tõttu kõrvale ning selle edasiarendamine lõpetatakse.

Käesoleva bakalaureusetöö koostamise raames läbiviidud arendusraamistike võrdluse põhjal asub autor seisukohale, et Tsolutions OÜ arendusvahendite virna juba kuuluva Aurelia arendusraamistiku kasutamine uue KMR-i arendamisel on kõige mõistlikum ja loogilisem valik.

Võrreldud alternatiividel on eeliseid, mis võivad pakkuda väärtust teistsugust funktsiooni täitvate üheleherakenduste puhul. Näiteks Reacti virtuaalne DOM, mis aitab optimeerida vaate elementide renderdamist mudeli andmete muutumise korral, oleks autori hinnangul tõhus vahend näiteks aktsiahindade jälgimiseks kasutatava veebirakenduse välja töötamisel. KMR puhul ei ole aga kuvatavad loendid nii mahukad ning *backendi* poolel toimuvad andmemuudatused nii sagedased, et selline renderdamist optimeeriv lahendus lisandväärtust looks.

Angulari kasutamise osas jäi autor skeptiliseks ennekõike erialakirjanduses korduvalt väljatoodud praktikute seisukohtade põhjal, mis viitasid arendusraamistiku jäikusele ja monoliitsusele. Mitmes kriitilisema alatooniga blogipostituses käis läbi hinnang: *„ainus viis Angularis midagi toimima saada on teha seda Angulari viisil (The only way to make thing work is the Angular way)“* [48]. Samuti jäi kõlama hinnang Angulari järsu õppimiskõvera osas.

Olukorras, kus T solutions OÜ arendusmeeskond on juba omandanud esmased kompetentsid Aurelia raamistikus arendamise kohta, näib asjatu hakata KMR-i arendamisel kasutama mingit muud arendusraamistikku – eriti olukorras, kus nende eelised ei paku arvestatavat lisandväärtust.

7.2. Andmekihi liides – Tpileti API

Loodud prototüübi põhjal arendatav KMR luuakse erinevalt täna kasutusel olevast tarkvarast teenuspõhisel arhitektuuril. Lihtsustatult kirjeldades tähendab see, et KMR-il kui allsüsteemil puudub endal andmebaas ning kõikide operatsioonide teostamiseks vajalikud andmed päritakse üle rakendusliidese ehk API.

Teenuspõhise arhitektuuri peamised omadused (eelised) on [49]:

- **platvormist sõltumatus** - teenuspõhine arhitektuur võimaldab omavahel integreerida teenuseid sõltumata programmeerimiskeelest, platvormist või äriprotsessidest;
- **nõrk sidestus** – teenused ei ole seotud konkreetse teenuse kasutajaga ja sõltuvad vähe teistest teenustest. See lihtsustab omakorda teenuste taaskasutatavust;
- **teenuste kompositsioon** – olemasolevaid teenuseid omavahel kombineerides (taaskasutades) saab luua uut funktsionaalust pakkuvaid teenuseid;
- **isekirjelduvus** – süntaktiliselt korrektne ning hästi kirjeldatud teenusekirjeldus on nii masin- kui ka inimloetav;

- **koosvõime** – organisatsioonisisese informatsiooni ja funktsionaalsuse avamine teistele osapooltele ning teenuste taaskasutamine muudab osapooled koosvõimelisteks.

T solutions piletimüügiplatvormi API on realiseeritud REST veebiteenustena. Tegemist on veebiteenustega, mis kasutavad standardseid HTTP *(Hypertext Transfer Protocol)* protokolli käske – GET, POST, harvemal juhul DELETE, PUT ning mille väljakutsutav funktsioon on määratud URI-ga *(Uniform Resource Identifier)*. REST veebiteenuste oluliseks tunnuseks ja nende jõudluse seisukohalt vaadates ka eeliseks on see, et REST teenuste puhul ei hoita teenusele vastava ressursi seisundit füüsiliselt rakendusserveris, vaid seisund sisaldub teenuseid realiseeriva rakenduse ja klientrakenduse poolt vahetatavate sõnumite sisus [50].

REST teenuste eeliseks on ka nende kasutamise lihtsus. Testimise eesmärgil võib lihtsamaid REST teenuseid välja kutsuda tavalises veebibrauseris, määrates aadressiks teenuse URI. Samuti on REST teenuste väljakutsumine võrreldes muude enamlevinud veebiteenuste standarditega lihtsam ka programmikoodist. Nagu eelnevalt mainitud on KMR-i arendamiseks kasutatavas Aurelia arendusraamistikus REST teenuste asünkroonseks väljakutsumiseks loodud vastavad meetodid ning raamistiku dokumentatsioonis esitatud nende kasutamist lihtsustavad koodinäited.

TG rakendas REST teenustel põhineval API-l baseeruvat teenuspõhist arhitektuuri esmakordselt enda portfelli kuuluva Tpileti mobiilirakenduse juures. Kuna ettevõtte erinevates müügikanalites toimub piletimüük üldjoontes sama äriloogika alusel, siis on mobiilirakenduse jaoks loodud REST teenuste komplekt taaskasutatav ka loodava KMR-i puhul.[1] Tpileti API dokumentatsioon on kättesaadav aadressil: https://www.tpilet.ee/api/Help/FullApi .

KMR-i toimimine API-t tarbiva klientprogrammina tähendab sisuliselt seda, et iga kasutaja poolt tehtavale operatsioonile järgneb päring mingi REST teenuse poole. Need päringud võivad olla:

[1] Bakalaureusetöö koostamise ajal läbiviidava T solutions piletimüügiplatvormi arhitektuurimuudatusega muudetakse oluliselt süsteemi andmemudelit, kuid olemasolevate veebiteenuste liidese spetsifikatsioon on võimalik säilitada kohandades teenuseid realiseerivaid rakenduskihi meetodeid uuele andmestruktuurile vastavalt.

- järgmise operatsiooni teostamiseks vajalike andmete hankimine – GET päring. Näiteks KMR-is kasutaja poolt sisestatud lähte- ja sihtpeatuse ning reisi kuupäeva alusel teostava otsingu käivitamisel esitab KMR päringu `GET timetable/bydate/{departureId}/{destinationId}/{date}` ning saab vastuseks JSON formaadis loendi käigusolevatest reisidest koos hinnainfo ja vabade kohtade arvuga. [51]
- eelmise operatsiooni tulemusena klientprogrammis tekkinud andmete või andmemuudatuste *backendile* edastamine. POST või DELETE päring. Näiteks KMR-is mingi reisi pileti müügiks märkimiseks kutsutakse välja päring `POST trip/book/{tripId}/{departureId}`, mis lisab pileti ostukorvi ning tagastab ostukorvile viitavat sessiooni ID-d sisaldava vastuse. [51]

Loodava KMR-i ja Tpileti API vahelise andmevahetuse illustreerimiseks on bakalaureusetöö lisas 2 toodud voogdiagramm, mis kirjeldab piletimüügi põhiprotsessi ulatuses, millise KMR-i operatsiooni tulemusena kutsutakse välja sellele operatsioonile vastav API meetod.

Tpileti mobiilirakenduse funktsionaalsus on võrreldes loodava KMR-ga siiski mõnevõrra kitsam. Seetõttu ei kata olemasolev API kogu KMR-i funktsionaalsuse toimimiseks vajaminevaid teenuseid. Puuduolevad teenused on seotud piletite otsimise ja muutmise ning müüdavate lisateenuste loendi pärimisega. Vastavate teenuste spetsifitseerimiseks annab sisendi käesoleva bakalaureusetöö raames loodud prototüüp, mille alusel on üheselt tuvastatav millist andmekomplekti on iga kasutaja poolt sooritatava operatsiooni järel vaja API-lt pärida.

7.3. Perifeersete riistvaraseadmete liidesed

Loodava KMR-i veebipõhise üheleherakendusena kavandamisel oli oluliseks küsimuseks see, kuidas lõimida perifeersed riisvaraseadmed nii, et nad toimiksid sama sujuvalt kui hetkel kasutusel oleva desktop rakenduse puhul. KMR-i perifeerseteks riistvaraseadmeteks on kaardimakseterminal ja piletite termoprinter. Nende seadmete sujuv lõimimine oli üks kaalutlusi miks 2010. aastal praegu kasutusel olevat KMR-i kavandades otsustati see realiseerida desktop rakendusena. Kuigi riistvaraseadmete tarkvaratehniline integratsioon ei ole üldjuhul infosüsteemide analüüsietapi osaks, oli

käesoleva projekti raames oluline kaardistada võimalikud lahendused, et saavutada kindlus kavandatava tarkvara üheleherakendusena realiseerimise võimalikkuse osas.

7.2.1. Kaardimakseterminal

Kaardimakseterminalidena kasutatakse TG ja tema frantsiisivõtjate piletimüügipunktides Ingenico Telium2 tooteperekonna kaardimakseterminale, mida rendivad kaupmeestele kõik Eesti pangad. Ingenico Telium2 tooteperekonna makseterminalide maaletooja on AS Hansab, mille tütarettevõte Ellore OÜ pakub makseterminalidega seonduvaid tarkvarateenuseid. Kaardimaksetерminali KMR-iga lõimimise võimaluste uurimiseks pidas autor konsultatsioone Ellore OÜ esindajatega.

Ingenico Telium2 kaardimaksetерminalide seadme sisesesse tarkvarasse on Eesti tarkvaraettevõtte Voicecom OÜ poolt välja arendatud maksesüsteemide integreerimise liides *posXML*. Tegemist on standardile PA-DSS[1] vastava tarkvaralahendusega, mille kasutamisel ei pea maksesüsteemide arendajad tegelema EMV *(Europay, MasterCard, and Visa)*[2] spetsiifiliste nõuetega ja läbi tegema EMV sertifitseerimist. [52]

PosXML võimaldab kassatarkvaral (antud kujul KMR-il) suhelda makseterminaliga TCP/IP protokolli alusel XML vormingud HTTP pakettide vahetamise teel. Makseterminalile on kassatarkvaraga samas arvutivõrgus omistatud kindel IP aadress, mille poole kassatarkvara sõnumivahetuse käigus pöördub.

PosXML-i kasutav klientprogramm on iga operatsiooni algatajaks saates välja sellele operatsioonile vastava XML vormingus sõnumi (näiteks kaardimakse algatamine). Makseterminal vastab sellele samuti XML vormingus vastussõnumiga. Vastus sisaldab operatsiooni tulemusele viitavat koodi ja sõltuvalt päringust võib sisaldada ka vastussõnumi sisu. Näiteks kaardimakse algatamiseks kassamüügitarkvara poolt väljasaadetava sõnumi (HTTP päringu) struktuur on järgmine: [52]

[1] Payment Application Data Security Standard – rahvusvaheline turvastandard, mille on loonud Kaardimaksete valdkonna turvastandardite nõukogu (Payment Card Industry Security Standards Council) [54]

[2] Maksekaartide ja makseterminalide tehniline standard, mida haldab EMVCo – kuu rahvusvahelise makselahenduse pakkuja poolt asutatud stadardi- ja sertifitseerimiseorganisatsioon.

Sõnumi päis:
```
http/1.0
content-length:185
content-type: text/xml
```

Sõnumi sisu:
```
<?xml version="1.0" encoding="UTF-8" ?>
<PosXML version="7.2.0">
      <TransactionRequest>
            <Amount>10000</Amount>
            <CurrencyName>EUR</CurrencyName>
      </TransactionRequest>
</PosXML>
```

Komponentide vahelise integratsiooni oluliseks eeltingimuseks on stabiilne side kassatarkvara ja makseterminali vahel. Selle jälgimiseks sisaldab *posXML ping* käsku, mida on võimalik kasutada ühenduse olemasolu regulaarseks kontrollimiseks. Sideühenduse kontrollimiseks saadab kassatarkvara välja kuueteistkümnendsüsteemi baidi 05 (ASCII ENQ[1]), millele makseterminal vastab ühenduse olemasolul baidi 06 (ASCII ACK[2]) tagastamisega. [52]

PosXML liidese abil on Ellore OÜ kinnitusel lõimitud Eestis sadu kassatarkvara süsteeme. Kuna liides on platvormist sõltumatu, ning suhtlus toimub harilike HTTP sõnumite vahetamise teel, siis on selle näol autori hinnangul tegemist väga sobiva lahendusega veebipõhise KMR-i ja makseterminali vahelise integratsiooni loomiseks.

7.2.2. Piletite termoprinter

Piletiprinteritena kasutatakse TG ja tema frantsiisivõtjate piletimüügikassades EPSON TM t88v termoprintereid. Seda tüüpi printerite ning suure valiku muud tüüpi termopriterite poolt kasutatava ESC/POS[3] käsustiku veebirakendustest välja kutsumiseks on loodud vabavaraline komponent nimega jZebra. Tegemist on Java appletiga, mida käitatakse veebilehe taustal ning mida on võimalik juhtida otse veebilehelt Javascripti kasutades. jZebra appleti kasutamiseks tuleb veebirakenduse HTML päises defineerida

[1] ASCII süsteemi baidi väärtus ühenduse olemasolu kontrollimiseks
[2] ASCII süsteemi baidi väärtus ühenduse olemasolu kinnitamiseks
[3] ESC – Epson Standard Command- Printeritootja Epson poolt välja töötatud printeri käsustiku standard.

viide appleti importimiseks. Sõltuvalt .jar faili asukohast failisüsteemis on appleti importimiseks vajaminev viide selline [53]:

```
<applet          name="jzebra"          code="jzebra.PrintApplet.class"
archive="<?=base_url()?>jZebra/dist/jzebra.jar"              width="10"
height="10">
      <param name="printer" value="zebra">
    </applet>
```

Appletile edastatatavad käsud kutsutakse välja veebirakenduse Javascripti koodist. Näide ühe tekstireaga printimisoperatsiooni välja kutsumisest on järgmine [53]:

```
function print_content(order_id){
         var html_to_print='';
          document.jzebra.append("SEE TEKST ON PRINDITUD JZEBRAGA \n");
            document.jzebra.print();
}
```

Autori hinnagul pakub jZebra applet väga lihtsat ja tõhusat meetodit piletiprinteri lõimimiseks veebipõhise tarkvaraga ning aitab lahendada piletiprinteri kui perifeerse riistvara KMR-st juhtimisega seotud probleemi.

8. Kokkuvõte

Käesoleva töö eesmärgiks oli töötada välja sõidupiletite kassamüügiks ettenähtud veebipõhise tarkvara prototüüp. Kavandatav rakendus asendab T grupp AS poolt praegu kasutavat KMR-i, mille väljavahetamise vajadus on tingitud ühelt poolt seni kasutusel olnud KMR-i funktsionaalsetest vajakajäämistest aga ka asjaolust, et ettevõtte piletimüügiplatvormi jõudluse parandamiseks on vaja läbi viia terviksüsteemi arhitektuurimuudatus, mille tulemusena juurutatakse teenuspõhine süsteemiarhitektuur.

Bakalaureusetöö tulemusena töötati välja uue KMR-i kogu funktsionaalsust detailselt imiteeriv prototüüp – nn täisfunktsionaalne prototüüp, mis on aluseks 2017. aasta teises pooles läbiviidavale arendusele. Seega on töö näol tegemist süsteemianalüüsiga ning töö tulemina valminud prototüüp on arenduse lähteülesande olulisim osa. Bakalaureusetöö koostamisel läbitöötatud erialakirjanduse ja autori isikliku kogemuse baasilt saab väita, et prototüüpimisel põhinev analüüs on efektiivne meetod, mille abil saab vähendada tavapärasel nõuete spetsifitseerimisel põhinevale analüüsiprotsessile omaseid probleeme.

Prototüüpimisel põhineva analüüsi eeliseks on ka see, et juba süsteemi kavandamise faasis on võimalik süvitsi tegelda loodava tarkvara kasutatavuse arendamisega. Kasutatavus määrab selle kuivõrd tõhusalt on tarkvara võimalik ettevõtte äriprotsesside läbiviimiseks kasutada, mistõttu on sellel otsene mõju äritegevuse tulemuslikkusele. Arendatava KMR-i kasutatavuse tagamiseks võttis autor prototüübi loomisel üle mitmeid kasutatavuse arendamise parimaid praktikaid ning viis enne lõpliku prototüübi valmimist läbi kasutatavuse testimise süsteemi tulevaste kasutajate peal. Testimine andis nii konkreetset sisendit kasutatavust parendavate muudatuste sisse viimiseks kui ka tõendas prototüüpimise ning kasutatavuse testimise olulist *frontend* arendusprojektides.

Käesoleva töö skoopi mahtus ka veebipõhiste üheleherakenduste arendamist lihtsustavate Javascripti arendusraamistike uurimine ning selle põhjal arenduseks kasutatavate vahendite osas nägemuse välja kujundamine. Antud teemakäsitlust pidas autor oluliseks seetõttu, et omandada isiklikku kompetentsi, mis annaks veendumuse, et arendusraamistike abil on võimalik luua uus KMR, mille kasutatavus ei jää alla *desktop*

rakenduse kasutusmugavusele. Samal eesmärgil kaardistas autor töö raames ka võimalused veebipõhise kassatarkvara ja perifeersete riistvaraseadmete lõimimiseks.

Bakalaureusetöö tulemusena loodud KMR-i prototüübi puhul õnnestus likvideerida seni kasutusel olnud tarkvarale omased funktsionaalsed puudujäägid ja edukalt testida tulemi toimimist süsteemi tulevaste kasutajate peal. Töö tulemil oli nii T grupp AS eesmärke kui ka autori professionaalset eneseteostust arvestades konkreetne praktiline väärtus kuivõrd see on sisendiks uue äritarkvara arendusele.

Summary

The purpose of this thesis was to create a prototype for a software which is designated for public transport ticket sales through ticket offices. The intended application shall substitute POS application which is currently used in by TG. Replacement of current software is on one hand driven by functional deficiencies but also by a fact that in order to improve the performace of company's ticket sales platform a comprehensive system architecture enhancement will be carried out and a service based architecture will be implemented.

As a result of current bachelor thesis a prototype imitating the functionality of POS application to be developed was created. This prototype shall be an input for a development process carried out in the second half of year 2017. Thus the outcome of this thesis is a system analysis and the created prototype is an essential input for the development. Based on literary sources and author's own experience the protype driven analysis is an effective method that enables to diminish the emergence of problems which are characteristic to conventional analysis process focusing only on specification of requirements.

The advantage of prototype driven analysis is also the possibility to focus in detail on software usability issues already in analysis phase. The usability determines how efficiently the software can be exploited to carry out company's business processes. Therefore it has a direct influnce on business productivity and profitability. To achive desired usability of intended POS application the author adapted benchmarking usability techiques and carried out usability testing on prospective users prior to completing the final version of the prototype. The testing gave an input for certain usability improvements as well as proof on how important prototyping and usability testing are in frontend based development project.

The scope of current thesis also included a study on Javascript frameworks designed for facilitating the development of single page applications as well as constructing a vison on which development tools and components should be used for the impending

development.The approach to this subject was important to aquire competences providing assertion that by implementing frontend frameworks it is possible to create web-based POS software with a usability that does not fall below desktop software. For the same purpose the author also mapped technical possibilities to integrate web-based POS software with peripheral hardware devices.

The prototype developed in the course of this bachelor thesis describes solution for problems that are characteristic to currently used POS software. The outcome of this thesis has a firm practical value considering both – the authors professional fulfilment and business goals of TG. The prototype shall be used an input for impending software development.

Kasutatud kirjadus

[1] S. Shaaban, „Web Based vs. „Desktop“ Software**“,** [Võrgumaterjal]. Aadress: https://nurelm.com/web-based-vs-desktop-software. [Kasutatud 20.03.2017].

[2] T. Marston, „What is the 3-Tier Architecture?", 14.10.2012. [Võrgumaterjal]. Aadress:http://www.tonymarston.net/php-mysql/3-tier-architecture.html#example-business-rules. [Kasutatud 14.05.2017].

[3] „What is backend as a service", Backendless Corp., [Võrgumaterjal]. Aadress: https://backendless.com/what-is-backend-as-a-service/. [Kasutatud 13.05.2017].

[4] N. M. Josuttis, SOA in practice, O' Reilly Media Inc. , 2007.

[5] T. Kotka, „Spetsifitseerimine vs prototüüpimine: lahendus kollase kassi probleemile“, 2013. [Võrgumaterjal]. Aadress:https://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwi2qePEqfTTAhXGAxoKHXM6CbYQFgggMAA&url=http%3A%2F%2Fdspace.ut.ee%2Fhandle%2F10062%2F32962&usg=AFQjCNFt0pnf33PG5U30HatJS-v9Fp7i9g. [Kasutatud 28. 03 2017].

[6] C. House, „The Joy of Prototype Driven Development“, 22.05.2012. [Võrgumaterjal]. Aadress: http://www.bitnative.com/2012/05/22/prototype-driven-development/. [Kasutatud 14.04.2017].

[7] „The 7 Best Prototyping Tools for UI and UX Designers in 2016“, prototypr.io, 19.04.2016. [Võrgumaterjal]. Aadress: https://blog.prototypr.io/the-7-best-prototyping-tools-for-ui-and-ux-designers-in-2016-701263ae65e8. [Kasutatud 15.04.2017].

[8] E. Ries, „The Prototype Mindset“, 08.03.2016. [Võrgumaterjal]. Aadress: https://medium.com/galleys/the-prototype-mindset-396c979a356f. [Kasutatud 16.04.2017].

[9] M. Nemberg, „Mis on kasutatavus, kasutajakogemus ja kasutatavuse inseneeria – UX algajatele“, 03.04.2015. [Võrgumaterjal].
Aadress: http://www.trinidad.ee/et/blogi/mis-on-kasutatavus-kasutajakogemus-ja-kasutatavuse-inseneeria-ux-algajatele/. [Kasutatud 21. 04. 2017].

[10] W. Queensbery, „Using the 5Es to understand users“, [Võrgumaterjal]. Aadress: http://www.wqusability.com/articles/getting-started.html. [Kasutatud 21.04.2017].

[11] „Ajax (programming)**“,** Wikipedia entsüklopeedia, [Võrgumaterjal]. Aadress: https://en.wikipedia.org/wiki/Ajax_(programming). [Kasutatud 20.04.2017].

[12] E. Wong, „Principle of Consistency and Standards in User Interface Design**“,** 04.2017. [Võrgumaterjal]. Aadress: https://www.interaction-design.org/literature/article/principle-of-consistency-and-standards-in-user-interface-design. [Kasutatud 17.04.2017].

[13] „Usage statistics and market share of Bootstrap for websites**“,** W3Techs, [Võrgumaterjal]. Aadress: https://w3techs.com/technologies/details/js-bootstrap/all/all. [Kasutatud 24.04.2017].

[14] M. Cossey, „Why Transitions Are Important**“,** 28.02.2012. [Võrgumaterjal]. Aadress: https://www.smashingmagazine.com/2012/02/mission-transition/ . [Kasutatud 23.04.2017].

[15] C. Bank ja J. Cao, The Guide to Usability Testing, UXPIN, 2016.

[16] „Usability testing**“,** US. Departnemt of Health and Human Services, [Võrgumaterjal]. Aadress: http://www.usability.gov/how-to-andtools/methods/usability-testing.html. [Kasutatud 08.05.2017].

[17] J. Nielsen, „Why You Only Need to Test with 5 Users**“,** Nielsen Norman Group, 19.03.2000. [Võrgumaterjal]. Aadress: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. [Kasutatud 10.05.2017].

[18] S. Krug, Rocket Surgery Made Easy, 2010.

[19] J. Cao, „Choosing Your Usability Tests and Participants**“,** 05.02.2015. [Võrgumaterjal]. Aadress: https://www.sitepoint.com/choosing-usability-tests-participants/. [Kasutatud 09.05.2017].

[20] J. Nielsen, „Interviewing Users“, Nielsen Norman Group, 26 07.2010. [Võrgumaterjal]. Aadress: https://www.nngroup.com/articles/interviewing-users/. [Kasutatud 12.05.2017].

[21] B. Parmar, „What is a web app? Pros & Cons of using them“, [Võrgumaterjal]. Aadress: https://www.imobdevtech.com/Blog/what-is-a-web-app-pros-cons. [Kasutatud 23.04.2017].

[22] „About Node.js“, Node.js Foundation, [Võrgumaterjal]. Aadress: https://nodejs.org/en/about/. [Kasutatud 17.03.2017].

[23] „Pros And Cons Of Using Frameworks“, Web Frameworks, [Võrgumaterjal]. Aadress: https://1stwebdesigner.com/web-frameworks/. [Kasutatud 25.04.2017].

[24] R. Strahl, „The Rise of JavaScript Frameworks - Part 1“, [Võrgumaterjal]. Aadress: https://weblog.west-wind.com/posts/2015/jul/18/the-rise-of-javascript-frameworks-part-1-today. [Kasutatud 25.04.2017].

[25] J. Tokareva, „Top 8 Best JavaScript Frameworks for 2017“, Techadmin.net, [Võrgumaterjal]. Aadress: https://tecadmin.net/top-8-best-javascript-frameworks-for-2017/#. [Kasutatud 01.05.2017].

[26] R. Morgan, „30 JavaScript frameworks and libraries to check out in 2017“, [Võrgumaterjal]. Aadress: https://getflywheel.com/layout/javascript-frameworks-libraries-2017/. [Kasutatud 01.05.2017].

[27] E. Räni, „JavaScripti raamistike võrdlus“, Tartu Ülikool, 2016. [Võrgumaterjal]. Aadress: Tartu Ülikooli Bakalaureusetöö, 2016, https://www.google.ee/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwjJ0_D1vtDTAhVBDiwKHeHWBHMQFggyMAI&url=https%3A%2F%2Fcomserv.cs.ut.ee%2Fhome%2Ffiles%2Fr2ni_informaatika_2016.pdf%3Fstudy%3DATILo. [Kasutatud 29.04.2017].

[28] „Top frameworks - Javascript“, HotFrameworks, [Võrgumaterjal]. Aadress: https://hotframeworks.com/languages/javascript. [Kasutatud 29.04.2017].

[29] G. Vestenberg, „Angular raamistik“, IT kolledži õppematerjal, [Võrgumaterjal]. Aadress: https://www.google.ee/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwjJ5ruJnc_TAhVGQJoKHXfwBfcQFggwMAM&url=http%3A%2F%2

Fenos.itcollege.ee%2F~mamangus%2Fi244%2Fesitlused%2FAngular.pptx&usg=AFQjCNEGuTxlNqr0C3qleA-QlcJ33diWkw&sig2=0Y3B0ua. [Kasutatud 29.04.2017].

[30] „Trending in open source“, GitHub, [Võrgumaterjal]. Aadress: https://github.com/trending. [Kasutatud 27.04.2017].

[31] K. Tammekivi, „Eesrakenduste loomise töövoo parendamine“, Tallinna Ülikool, 2014. [Võrgumaterjal]. Aadress: http://www.cs.tlu.ee/teemad/get_file.php?id=310. [Kasutatud 05.04.2017].

[32] „React vc Angular: Two sides of Javascript“, Cleveroad, 02.2017 [Võrgumaterjal]. Aadress: https://www.cleveroad.com/blog/react-vs-angular-ultimate-performance-research-2017. [Kasutatud 04.05.2017].

[33] E. McCormick, „Angular 2's Learning Curve“, 06.07.2016. [Võrgumaterjal]. Aadress: https://edm00se.io/web/angular-2-learning-curve/. [Kasutatud 05.05.2017].

[34] A. Alexseyenko, „Angular 2 vs React. What to chose in 2017?“, 23.02.2017. [Võrgumaterjal]. Aadress: http://blog.techmagic.co/angular-2-vs-react-what-to-chose-in-2017/. [Kasutatud 04.05.2017].

[35] M. Stoiber, „Introduction to React.js“, [Võrgumaterjal]. Aadress: http://academy.plot.ly/react/1-introduction/. [Kasutatud 04.05.2017].

[36] „Developer Survey Results 2016“, StackOverflow portaal, [Võrgumaterjal]. Aadress: http://stackoverflow.com/insights/survey/2016#community. [Kasutatud 01.05.2017].

[37] M. Fowler, „InversionOfControl“, 05.06.2005. [Võrgumaterjal]. Aadress: https://martinfowler.com/bliki/InversionOfControl.html. [Kasutatud 05.05.2017].

[38] P. Vallejo, „Beginner's introduction to React.js“, 20.02.2015. [Võrgumaterjal]. Aadress: https://axiacore.com/blog/beginners-introduction-reactjs/. [Kasutatud 05.05.2017].

[39] S. Kamani, „React is a framework“, 16.11.2016. [Võrgumaterjal]. Aadress: http://www.sohamkamani.com/blog/2016/11/16/react-is-a-framework/. [Kasutatud 05.05.2017].

[40] „React: The Virtual DOM - Fighting Wasteful DOM Manipulation“, Codeacademy, [Võrgumaterjal]. Aadress: https://www.codecademy.com/articles/react-virtual-dom. [Kasutatud 05.05.2017].

[41] A. Ray, „ReactJS For Stupid People“, [Võrgumaterjal]. Aadress: http://blog.andrewray.me/reactjs-for-stupid-people/. [Kasutatud 05.05.2017].

[42] A. Mikhalchenko, „Aurelia vs. Angular 2 — A Code Comparison“, [Võrgumaterjal]. Aadress: https://www.toptal.com/angular-js/aurelia-vs-angular-2. [Kasutatud 06.05.2017].

[43] „5 Benefits Of Choosing Aurelia js Over AngularJS“, Valuecoders portaal, [Võrgumaterjal]. Aadress: http://www.valuecoders.com/blog/technology-and-apps/aurelia-js-framework-review-typescript-router/ . [Kasutatud 05.05.2017].

[44] „Introduction to Aurelia“, GitBooks.io, [Võrgumaterjal]. Aadress: https://straightforward-aurelia.gitbooks.io/straightforward-aurelia/content/chapter1.html. [Kasutatud 06.05.2017].

[45] J. Duffy ja J. Brown, „Aurelia: An Introduction“, Code Magazine, [Võrgumaterjal]. Aadress: http://www.codemag.com/Article/1607091 . [Kasutatud 07.05.2017].

[46] R. Eisenberg, „What is Aurelia?“, Durandal Inc, [Võrgumaterjal]. Aadress: http://aurelia.io/hub.html#/doc/article/aurelia/framework/latest/what-is-aurelia/1. [Kasutatud 06.05.2017].

[47] M. Burgess, „All JavaScript frameworks are terrible: A case study in confirmation bias and goalpost shifting“, 05.02.2017. [Võrgumaterjal]. Aadress: https://medium.com/@mattburgess/all-javascript-frameworks-are-terrible-e68d8865183e. [Kasutatud 07.05.2017].

[48] M. Moise, „AngularJS: The Bad Bits“, [Võrgumaterjal]. Aadress: https://www.thoughtworks.com/insights/blog/angularjs-bad-bits. [Kasutatud 05.05.2017].

[49] „Teenusepõhise arhitektuuri omadused“, Riigi Infosüsteemi Amet, 24.01.2017. [Võrgumaterjal]. Aadress: https://moodle.ria.ee/mod/page/view.php?id=445. [Kasutatud 13.05.2017.].

[50] E. Lenk, „Pädevushaldus RESTful veebiteenuste abil“, Tallinna Ülikool, 2010.

[51] „TSolutions API documentation“, T solutions OÜ, [Võrgumaterjal]. Aadress: https://www.tpilet.ee/api/Help. [Kasutatud 10.05.2017].

[52] „PosXML Protocol Specification“, Voicecom OÜ, 2013.

[53] A. Thompson, „How to print in reciept printer with jzebra applet“, 19.08.2013. [Võrgumaterjal]. Aadress: http://stackoverflow.com/questions/18307877/how-to-print-in-reciept-printer-with-jzebra-applet. [Kasutatud 12.05.2017].

[54] „PA-DSS“, Wikipedia ensüklopeedia, [Võrgumaterjal]. Aadress: https://en.wikipedia.org/wiki/PA-DSS. [Kasutatud 12.05.2017].

[55] „ECMAScript“, Wikipedia entsüklopeedia, [Võrgumaterjal]. Aadress: https://en.wikipedia.org/wiki/ECMAScript. [Kasutatud 02.05.2017].

[56] „Write once, run anywhere“, Wikipedia entsüklopeedia, [Võrgumaterjal]. Aadress: https://en.wikipedia.org/wiki/Write_once,_run_anywhere. [Kasutatud 07.04.2017].

[57] „X-tee liideste arendajate koolitusmaterjalid“, Riigi Infosüsteemi Amet, 24.01.2017. [Võrgumaterjal]. Aadress: https://moodle.ria.ee/mod/page/view.php?id=443. [Kasutatud 13.05.2017.].

[58] „TypeScript“, Wikipedia entsüklopeedia, [Võrgumaterjal]. Aadress: https://en.wikipedia.org/wiki/TypeScript. [Kasutatud 04.05.2017].

LISA 1 - Testimise kaaskiri

Hea kolleeg

Kindlasti oled märganud, et kassamüügiprogramm, mida enda töös igapäevastelt kasutad, on muutunud üsna aeglaseks ning selle abil ei ole võimalik teostada kõiki toiminguid, mida kliendid meilt küsivad. Kuigi teie teete klienditeenindajatena igapäevaselt suurepärast tööd, on kassamüügiprogrammi võimekus hakanud kahjustama nii klienditeeninduse taset kui ka teenindajate töö mugavust. Ettevõtte juhtkond ja arendusmeeskond on seda probleemi teadvustanud ning tuvastanud, et pisimuudatustega seda lahendada ei ole võimalik. Sel põhjusel oleme algatanud Tpileti piletimüügisüsteemi suuremahulise uuenduse, mille käigus luuakse ka uus kassamüügiprogramm.

Tänaseks on valminud uue kassamüügiprogrammi prototüüp. Palun sinu abi selle testimisel, et saada ausat ja otsekohest tagasisidet, kas oleme arendusega õigel teel. Prototüüp ei ole kasutusvalmis töövahend vaid loodava programmi „makett“, mille ekraanivaated kirjeldavad võimalikult täpselt, kus uues programmis mingi nupp või andmete jaotis paiknema hakkab ja milliste klahvivajutuste ja hiireklikkide abil kassatöö toiminguid teha saab. Seega töötab prototüüp nagu päris programm ning täna palun sul testimise eesmärgil seda kasutada kahe tavapärase olukorra lahendamiseks.

Palun, et sa prototüüpi kasutades kõva häälega selgitaksid, mida sa näed, mida sa parajasti üritad teha või ekraanilt leida ning mida sa mõtled. Ära muretse selle pärast, et see mida sa kommenteerid võiks kõlada kohatu või näida rumal. Kõva häälega kommenteerimine võib sulle tunduda veidi totter, kuid see on väga suureks abiks, et saaksime luua sellise töövahendi, mida on võimalik kasutada mugavalt ja kiiresti.

Kõige olulisem, mida soovin rõhutada on see, et **me testime uut programmi mitte sind**. **Sinul ei ole testimise käigus võimalik mitte midagi valesti teha**. Kindlasti ei pea sa muretsema selle pärast et testi põhjal hinnataks mistahes moel sinu oskusi. Samuti ei pea sa muretsema selle pärast, et kas tegutsed piisavalt kiiresti. Võta aega, et ringi vaadata, kirjeldada mida sa näed ja mõtled ning tegutse endale mugavas tempos.

Samal ajal, kui sina tegutsed teen ma mõningaid märkmeid, kuid juhul kui sul tekib programmi kasutamisel mistahes küsimusi, siis võid neid minult kohe küsida. Ma ei pruugi saada kõikidele su küsimustele kohe vastata, sest tahame aru saada, kui hästi tulevad kasutajad toime, siis kui nad programmi iseseisvalt kasutaksid. Siiski on oluline, et sa küsimuste tekkimisel need ikka välja ütled, sest tekkivad küsimused võivad viidata sellele, et peaksime midagi muutma. Pealegi käime ka need küsimused, millele ma kohe vastata ei saa, testi lõppedes koos üle.

Olukorrad, mida palume sul prototüüpi kasutades lahendada on sellised, mille sarnastega oled suure tõenäosusega enda töös korduvalt kokku puutunud.

Esimese olukorra kirjeldus: *Klient soovib osta endale ja oma eelkooliealisele lapsele homme kell 14:00 väljuvale Tallinn – Tartu reisile pileteid. Klient soovib istuda bussis enda lapsega kõrvuti asetsevatel istmetel võimalusel bussi eesosas. Klient esitab ostmisel soodustuse saamiseks PINS kaardi numbriga 82452341 ning maksab sularahas 20€ rahatähega.*

Teise olukorra kirjeldus: *Klient helistab kassasse ja selgitab, et tal on homme kell 9:00 väljuvale Tallinn – Pärnu reisile pilet ostetud, kuid ta tuleb bussi peale Vana-Pääsküla peatusest Tallinnas. Kliendile bussis koha tagamiseks on vaja vormistada pileti muudatus ilma muutmise teenustasu kohaldamata. Kliendi nimi on Paul Puuraid.*

Ilmselt paned tähele, et mõlemad olukorrad sisaldavad midagi, mida praeguse müügiprogrammiga lahendada ei saa. Prototüübis on nende toimingute teostamiseks võimalused loodud. Palume sul mõlemat juhtumit lahendada kaks korda. Mõlemal sooritusel on ekraanivormide ülesehitus natuke erinev. Erinevate variantide läbikatsetamise kaudu üritame saada selgust, kumb lahendusvariant on piletimüügiprogrammi tulevaste kasutajate jaoks lihtsam ja loogilisem.

Tänan sind, et aitad Tpiletil uueneda

Ingmar Roos

LISA 2 – Piletimüügi protsessi andmevahetuse voogdiagramm