TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Märt Saarmets
176414IAAM

VISUAALSEL PROGRAMMEERIMISEL PÕHINEVA VEEBIRAKENDUSTE LOOMISE PLATVORMI ANALÜÜS JA KAVANDAMINE

Magistritöö

Juhendaja: Priit Raspel
MSc

Tallinn 2019

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: Märt Saarmets

14.05.2019

Annotatsioon

Käesoleva magistritöö eesmärgiks on viia läbi süsteemianalüüs platvormile, mis võimaldab selle kasutajatel lihtsasti luua enda tingimustele vastavaid spetsiifilisi pilvepõhiseid rakendusi kasutades selleks visuaalse programmeerimise meetodeid.

Käesolev magistritöö koosneb neljast osast. Töö esimeses pooles on kirjeldatud ülesandepüstitust ja töö eesmärki ning analüüsitud olemasolevat kirjandust.

Töö teises pooles on läbi viidud süsteemianalüüs, mille käigus koostati esialgne prototüüp, viidi läbi kasutajauuring ning püstitati nõuded. Analüüsi tulemusena kirjeldati loodava süsteemi arhitektuurilist lahendust, ärireegleid ning kasutusmalle.

Töö tulemusena on võimalik alustada analüüsitud veebiplatvormi esmase versiooni programmeerimisega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 61 leheküljel, 6 peatükki, 19 joonist, 4 tabelit.

Abstract

Analysis and Design of a Visual Programming Based Web Application Development Platform

Ordering software that suits end-user needs is very costly for the average user and is often not financially feasible. This problem can be solved by a platform that allows non-programmers to create their own cloud-based web applications using visual programming techniques. The purpose of this master's thesis is to conduct a system analysis of such a platform.

This master thesis consists of four parts. The first half of the thesis describes the task statement and the purpose of the work and analyzes the existing literature.

In the second half of the work, a system analysis was carried out, during which a preliminary prototype was created, a user survey was carried out and requirements were set. As a result of the analysis, the architectural design, business rules and use cases were described.

As a result of the user survey, 42% of the respondents would use such a platform. The survey showed also that 63% of respondents are willing to contribute 5 hours to 1 month to create new software that meets their needs. Thus, users are ready to invest enough time to learn how to use the platform and create the applications they need.

From a system architecture point of view, the platform applies the physical differentiation of applications and databases to ensure the security of the applications and to make it easy for individual applications to scale. The NoSQL databases of the web applications being created are located in the cloud service and a new instance is created for each application. The web application code is located in Docker containers and is distributed between different servers.

As a result of the work it is possible to start programming the first version of the analyzed web platform.

The thesis is in Estonian and contains 61 pages of text, 6 chapters, 19 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Program Interface* – rakendusliides võimaldamaks erinevate tarkvarade omavahelist suhtlust
Back-end	„Tagarakendus“. Klient/server süsteemides loetakse back-endiks tavaliselt serverit.
Beta	Testimisjärgus tarkvara versioon
Drag and drop	Hiirekursori abil elementide teisaldamine uude asukohta
Flow-chart	Vooskeem
GDPR	General Data Protection Regulation – Isikuandmete kaitse üldmäärus
LKP	Lõppkasutaja programmeerimine
Plugin	Pistikprogramm – moodul või komponent, mis annab tarkvarale lisafunktsionaalsust
WYSIWYG	Tekstiredaktor. "What you see is what you get" akronüüm.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Erinevate valmislahenduse kujul infosüsteemide ja tarkvarapakettide loomisel on arendaja poolt võetud tihti eesmärgiks, et toode peab sobima võimalikult suurele kasutajaskonnale. See on tingitud arendaja ärieesmärgist teenindada suurt hulka kliente ja seeläbi teenida maksimaalset võimalikku kasumit.

Selliste süsteemide puhul leidub alati kasutajaid, kellele teatud funktsionaalsus ei sobi või on midagi muud olulist puudu. Samas ei saa arendaja selliste kasutajate ettepanekuid tihti arvesse võtta, sest arvestada tuleb enamuse vajaduste ja heaoluga. Üksikute erandite tegemine on kallis ning muudab infosüsteemi arhitektuuri keerukaks ja suurendab vigade tõenäosust. Enda vajadustele sobiva tarkvara tellimine on aga tavakasutaja jaoks väga kulukas ning tihti ei ole äriliselt tasuv. Seda probleemi aitab lahendada platvorm, mille abil saavad programmeerimist mitte oskavad inimesed luua ise endale sobivaid ja vajalikke pilvepõhiseid veebirakendusi.

Käesolev magistritöö koosneb neljast osast. Töö esimeses osas on kirjeldatud ülesandepüstitust ja töö eesmärki. Teine osa analüüsib olemasolevat kirjandust, mille alusel saab teha loodava süsteemi analüüsi.

Töö kolmas osa on süsteemianalüüs, mille käigus koostatakse esialgne prototüüp, viiakse läbi kasutajauuring ning püstitatakse nõuded. Neljandas osas tuuakse välja loodava süsteemi arhitektuuriline lahendus, kirjeldatakse ärireegleid ning kasutusmalle.

Neljandas osas tehakse kokkuvõte ja esitatakse töö olulisemad tulemused ja järeldused.

Töö lisades on välja toodud esmase prototüübi ekraanivaated, kasutusmallid, platvormi andmebaasi füüsiline andmemudel ning selle mudeli tabelite semantika. Lisaks kahe olemasoleva platvormi ekraanipildid.

2 Ülesandepüstitus ja töö eesmärk

Hetkel leidub turul tööriistu, millega lõppkasutaja saab luua endale sobiva tarkvara, kuid valik on väike ning nende kasutamine keeruline. Endale sobiva tarkvara loomine tähendab seda, et kasutaja peab saama valida kõik tarkvara komponendid ning kirjeldada kogu rakenduse loogika. Need komponendid peavad olema seostatavad konkreetsete andmetega. Kasutamise lihtsus tähendab, et inimene, kes oskab kasutada kontoritarkvarapakettides olevat tabeltöötlustarkvara peab saama edukalt hakkama ka enda poolt loodava veebirakenduse andmete modelleerimisega. Seega peab kogu tarkvara loomise protsess olema võimalik ilma programmikoodi kirjutamata. Selle jaoks on aga vaja arendusplatvormi, mis võimaldab kasutajatel luua tarkvara visuaalse programmeerimise abil, pakkudes valmiskomponente ning visuaalset programmeerimist toetavat kasutajaliidest.

Käesoleva magistritöö eesmärgiks on viia läbi süsteemianalüüs platvormile, mis võimaldab selle kasutajatel lihtsasti luua enda tingimustele vastavaid spetsiifilisi pilvepõhiseid rakendusi. Töö käigus kaardistatakse kavandatava infosüsteemi nõuded ja tehakse ettepanekud tehniliste lahenduste osas. Töö ühe osana valmib ka planeeritava infosüsteemi prototüüp. Magistritöö tulemusel valmiva analüüsi ja projekti põhjal on võimalik alustada sellise arendusplatvormi programmeerimisega.

Loodava infosüsteemi kavandamisel tuleb tähelepanu pöörata järgnevatele tingimustele:

- Infosüsteemi kasutajal peab olema võimalik visuaalse programmeerimise abil luua enda nõuetele vastavaid veebirakendusi.
- Infosüsteemi kasutamine peab olema võimalik arvutis, millel on internetiühendus ja paigaldatud kaasaegne veebilehitseja. Veebilehitseja jaoks ei tohiks olla nõutud lisamoodulite paigaldamine.
- Tarkvara peab olema võimeline kasutama isik, kes ei ole varem tarkvaraarendusega kokku puutunud ning kellel puuduvad oskused mõne traditsioonilise programmeerimiskeele kasutamiseks.

Kuna tegemist on uue süsteemi loomisega siis puuduvad esialgsed nõuded ning need tuleb töö käigus välja selgitada. Lisaks vajavad lahendamist järgmised küsimused:

- Kas sellise platvormi järele on nõudlust?

- Kui palju on keskmine kasutaja nõus panustama aega uue tarkvara kasutama õppimisele?
- Kas on üldse mõistlik kasutada visuaalset programmeerimist?
- Kuidas tagada selliste rakenduste turvalisus?
- Kuidas tagada seda, et erinevatel rakendustel oleks alati piisavalt serveri ressursse?
- Kuidas tagada vastavus andmekaitsenõuetele?

Nimetatud tingimuste ja küsimuste lahendamiseks tuleb leida sobivad metoodikad ja läbi viia kasutajauuringud. Kasutajauuringute väljundiks on esialgsed nõuded, mille alusel kirjeldatakse uue infosüsteemi arhitektuuri. Süsteemi arhitektuuri kirjeldamiseks tuleb lisaks kasutajate nõuetele analüüsida ka kirjandust, et saada ülevaade analoogsete süsteemide tehnilistest lahendustest.

Eelnevalt nimetatud enda vajadustele vastava veebitarkvara loomiseks peab kasutaja saama arendusplatvormil selle kokku panna vastavatest komponentidest kasutades visuaalset programmeerimist. Visuaalse programmeerimise abil peab kasutajal olema võimalik näiteks valida nimekirjast endale sobiv komponent ning lohistada see hiirekursoriga ekraanil sobivasse asukohta. Nimetatud komponentideks võivad veebirakenduse puhul olla näiteks nupud, tekstiväljad, pildid, tabelid, failide üles- ja allalaadimise vormid jne. Sellised komponendid on eelnevalt arendusplatvormil olemas ning kasutaja saab neid lihtsasti enda rakenduses õigesse asukohta paigutada ning teha ka lihtsamaid muudatusi disainis – muuta värvi, teksti suurust jne.

Kasutajate autentimist nõudvate rakenduste jaoks peab olema lahendatud ka autentimisvõimaluste lisamine ja ligipääsude haldus. Lisaks peab kasutajal olema võimalik kirjeldada ning sisestada rakenduses kasutatavaid andmeid ja siduda neid eelnevalt kirjeldatud komponentidega. Andmete haldamine ja modelleerimine peab olema sarnane tabelitöötlustarkvarale kuna see on intuitiivne ja tõenäoliselt on suur osa arvutikasutajatest sellega eelnevalt kokku puutunud.

3 Kirjanduse ülevaade

Absoluutarvudes ei ole suur osa programmidest tänapäeval enam kirjutatud mitte tarkvaraarendajate poolt, vaid need on kirjutanud inimesed, kes on oma valdkonna spetsialistid ning vajavad neid programme enda ametialaste ülesannete toetamiseks. Siinkohal on programmina mõeldud ka näiteks tabelitöötlustarkvaras valemite ja muu funktsionaalsuse kasutamist, et koostada isearvutavaid raporteid, analüüse jne. Teise näitena võib tuua kasutajaliideste kujundaja, kes kasutab spetsiaalset tööriista algse prototüübi koostamiseks ja kliendile tutvustamiseks. [1, p. 1]

Need inimesed ei kasuta küll traditsioonilisi programmeerimisvõtteid ja -vahendeid, kuid peavad lahendama mitmeid sarnaseid probleeme võrreldes tarkvaraarendajatega – näiteks nõuete püstitamine, disainiotsuste tegemine, taaskasutuse küsimused, integratsioon teiste süsteemidega, testimine ja silumine. [1]

3.1 Lõppkasutaja programmeerimine

Ameerika Ühendriikide statistikaameti andmetel oli aastal 2012 riigis elukutselisi tarkvaraarendajaid veidi alla 3 miljoni, kuid inimesi, kes kasutavad tööalaselt tabelitöötlustarkvara ja erinevaid andmebaase andmete töötluseks üle 55 miljoni. Nende hulka kuuluvad ka inimesed, kes disainivad veebilehti, töötlevad suurandmeid, haldavad ettevõtete kodulehti, produtseerivad muusikat jne. Sellisel tasandil tarkvara loomist nimetatakse lõppkasutaja programmeerimiseks (*End-user programming, end-user development*) ehk lühendatult LKP (*EUD)* [1, p. 2] [2, p. 1]. See termin võeti Euroopa teadlaste poolt esmakordselt kasutusele aastal 2007 selle teema käsitlemiseks korraldatud konverentsil. Alates sellest ajast on seda teemat erinevatest aspektidest uuritud. Põhilised suunad on lihtsa programmeerimiskeele või -platvormi loomine, viisardipõhiste teenuste (*wizard-supported services*) integreerimine LKP vallas ja uuringud inimese-arvuti suhtluses ehk millisel kujul peaksid sellise platvormi komponendid olema, et keskmine inimene suudaks enda nõudmistele vastava tarkvara ka reaalselt valmis teha. [2]

Lõppkasutaja programmeerimist defineeritakse tavaliselt eesmärgipõhiselt, mitte töövõtete põhiselt. Seega defineeritakse lõppkasutaja programmeerimist kui programmi kirjutamist pigem enda tarbeks, kui avalikuks jagamiseks olenemata selle programmi

kirjutamise viisist (Java programmeerimiskeel, arvutustabel, pilditöötlustarkvara skriptid jne.). Professionaalse tarkvaraarenduse eesmärk on aga reeglina tarkvara kirjutamine kellelegi teisele [1, p. 4].

Lõppkasutaja programmeerimine võib tegevusena olla lihtsam, kuid tulemuse tähtsus ja tagajärjed on tihti võrreldavad klassikalise tarkvaraarenduse käigus arendatud tarkvara kasutamisega. Näiteks kaotas üks Texase naftaettevõte miljoneid dollareid vea tõttu arvutustabeli ühes valemis [1, p. 2]. Taoliste kvaliteediprobleemide tõttu on hakatud teaduslikult uurima lõppkasutaja programmeerimist, et töötada välja lahendusi, mis vähendaksid selliste vigade tõenäosust. [1, p. 2]

Uuringute põhjal on järeldatud, et lõppkasutajatele programmeerimisplatvormide ja -tööriistade loomisel ei piisa sellest, kui kasutajat harida üldiste tarkvaraarenduse printsiipide osas ja nende tööriistade kasutamisel. Tähtis on aru saada miks konkreetsed kasutajad tegelevad programmeerimisega ning mis on selliste programmide eesmärk. Oluline on leida põhjus mis motiveerib neid kasutajaid programmeerima mitte õpetada neile programmeerimist [1, pp. 34-35].

3.2 Vajadus lõppkasutaja programmeerimisplatvormide järele

Infotehnoloogia ja internetiühenduse areng ning laiem levik on loonud väike- ja keskmise suurusega ettevõtete jaoks võimalused konkureerida suurte korporatsioonidega ning konkurentidega globaalselt. Samal ajal on nende võimaluste kasutuselevõtmisel väikeettevõtete jaoks mitmeid probleeme – näiteks ei suuda tarkvaraarendaja luua täpselt sellistele nõuetele vastavat tarkvara nagu tellija soovib või ületab projekt eelarve, mida paljud väikeettevõtted endale lubada ei saa. [3, p. 1]

Põhilised probleemid, mis tekivad väike- ja keskmise suurusega ettevõtetel tarkvara arendamise käigus [3, pp. 2-3]:

1. Tarkvaraarendajad ei mõista päris täpselt, millised on tellija nõuded.
2. Tellijad ja kasutajad ei suuda täpselt määratleda ärist tulenevaid võimalikke muudatusi tulevikus ja tarkvaraarendajad ei suuda tekkinud muudatusi edukalt juba olemasolevasse tarkvarasse integreerida.

3. Aina suurenev vajadus uute veebirakenduste järele, et uute ärieesmärkide ja nendest tuleneva digitaliseerimisega graafikus püsida.
4. Vajadus hoida ajakohast infot veebirakenduses, et see oleks kättesaadav ka laiemale kasutajaskonnale.
5. Veebirakenduste arendusprojektid ületavad ettenähtud graafikut ja eelarvet.

Soov võtta kasutusele spetsiaalne tarkvara ei tulene ainult automatiseerimisest ega interneti kaudu mugavamast andmete jagamisest. Üheks põhjuseks võib olla ka inimese poolt tekitatavate vigade tõenäosuse vähendamine.

Erinevad uuringud ettevõtete juures on välja toonud, et suurem osa (86% - 100%) uuritud tabeltöötlustarkvarades loodud tabelitest sisaldavad vigu. Sealjuures ei ole vahet, kes need tabelid koostas – kas algaja või aastatepikkuse kogemusega töötaja, vigade esinemise tõenäosus on neil sama. [4, p. 2]

Vead, mis selliste tabelite kasutamisel tekivad on erinevad. Sealjuures kõige loogilisemana tunduv viga ehk numbri ebakorrektne sisestamine on üks haruldasemaid. Pigem tekivad vead andmete sisestamisel valesse lahtrisse, loogikavead algoritmi koostamisel ja andmete väljajätmine mudelist ehk mõnest algoritmist. Vead ei piirdu ka ainult tabeli koostamisel, vaid need tekivad ka hiljem selle kasutamise käigus – keegi sisestab andmed valesse lahtrisse ja mudelid ei tööta enam, kustutab või muudab mingeid valemeid jne. [4, pp. 3-4]

Selliste vigade esinemist toetab ka see, et organisatsiooni tasemel ei ole tabelitöötlustarkvarades koostatavad tabelid kuidagi reguleeritud ega kontrollitud. Lisaks aitab sellele kaasa ka inimese psühholoogia – ühe uuringu käigus küsiti selliste tabelite koostajatelt, kui tõenäoliselt nad enda arvates tegid kuskil vea. Mediaan-tulemus oli, et 10% vastajatest arvas enda tabelites olevat vigu. Tegelikult esines vigu 86% kasutajatest. Kui samade inimeste käest küsiti, kas nad arvavad end olevat pigem nende 14% hulgas, kes õigesti vastas, tõstsid üle poolte enda käed ehk olid siiski kindlad, et nemad ei teinud enda töös vigu. [4, p. 4]

Arvestades ettevõtete soovi enda tööprotsesse automatiseerida ja pakkuda klientidele ning töötajatele kaasaegseid lahendusi ning vähendada tabeltöötlustarkvaradest tekkivaid vigu võib järeldada, et need ettevõtted vajavad tööriistu, millega luua endale sobivat

tarkvara. Järgnevalt on välja toodud probleemid, mida autori arvates suudaks projekteeritav lõppkasutaja programmeerimist toetav platvorm lahendada:

1. Kuna konkreetses ettevõttes töötav inimene tunneb äriprotsesse ning kõiki erisusi tunduvalt paremini, kui tarkvaraarendaja on kõikide nõuete arvestamine programmeerimise käigus sujuvam.
2. Kui äriprotsessid või -eesmärgid muutuvad ei pea tarkvaras muudatuste tegemiseks ostma tarkvaraarendusteenust. Muudatuste sisseviimine on kiirem ja odavam.
3. Väikese aga spetsiifilise tarkvara loomine on odavam. Kui turul ei leidu täpselt vajaminevat tarkvarapaketti tuleks see iga kord lasta spetsiaalse tööprotsessi jaoks tarkvaraarendajatel luua.
4. Tarkvara loomiseks ei pea palkama eraldi inimest ega tellima seda teenusena. Sellega saab tegeleda inimene, kes on vastava protsessiga ettevõttes kõige rohkem kursis.
5. Andmete sisestamisel tekkivate vigade vähendamine – veebirakenduses saab kasutaja määrata millist tüüpi andmeid saab vormi sisestada ning valel kujul või vales vormingus andmete korral kuvatakse veateadet.
6. Valemitest tekkivate vigade tõenäosuse vähendamine – veebirakenduses ei kopeerita andmeid ja vorme sarnaselt tabeltöötlustarkvarade failidele. Andmete sisestamisel ei saa lõppkasutaja muuta valemeid. Kui tarkvara looja eksib mõne valemi koostamisel saab ta selle parandada kartmata, et kuskil võib levida koopia failist, kus sama viga veel sees on.
7. Kuna veebirakenduses saab määrata kasutajaid ja ligipääsuõiguseid on võimalik samade andmete põhjal genereerida erinevaid raporteid, tabeleid ja kuvada infot muul viisil vastavalt sellele, kes hetkel neid andmeid vaatab. Seega ei ole vaja koostada erinevaid tabeleid selleks, et organisatsiooni siseselt infot vaid selleks luba omavatele inimestele jagada.
8. Selline platvorm võimaldaks protsesside automatiseerimist ja lihtsustamist ka ettevõtetele, kes muidu ei oleks võimelised endale seda finantsiliselt lubama.

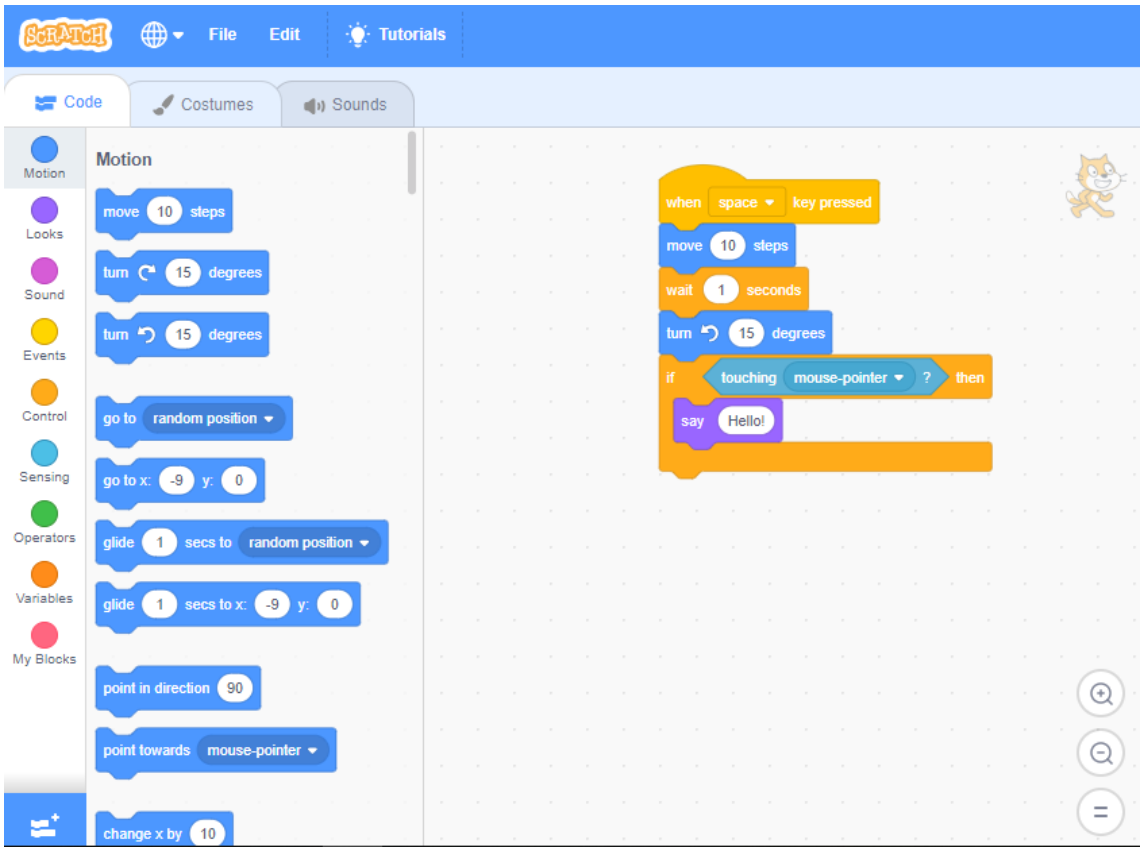
3.3 Visuaalne programmeerimine

Lõppkasutaja programmeerimise võimaldamiseks on loodud mitmeid lahendusi. Kõige levinumad on arvutustabelid. Teise näitena võib tuua spetsiaalsed skriptimiskeeled, mis võimaldavad kasutajal kirjutada laiendusi olemasolevale tarkvarale. Skriptide miinusena tuuakse välja suurt õppimiskõverat ning seda, et need ei ole kuigi veakindlad. [5]

Üks skripitmiskeelte variatsioone on loomuliku keele imiteerimine. Selle eesmärk on lihtsustada lõppkasutaja jaoks programmeerimist selliselt, et programmeerimiskeel oleks võimalikult sarnane inimese kõnekeelele. See meetod on teoorias paljulubav, kuid olemasolevad süsteemid on alles enda arengu algusfaasis ning nende arendamine sisaldab keeruliste probleemide lahendamist, mis ei pruugi olla enam projekti mõttes tasuv. [5]

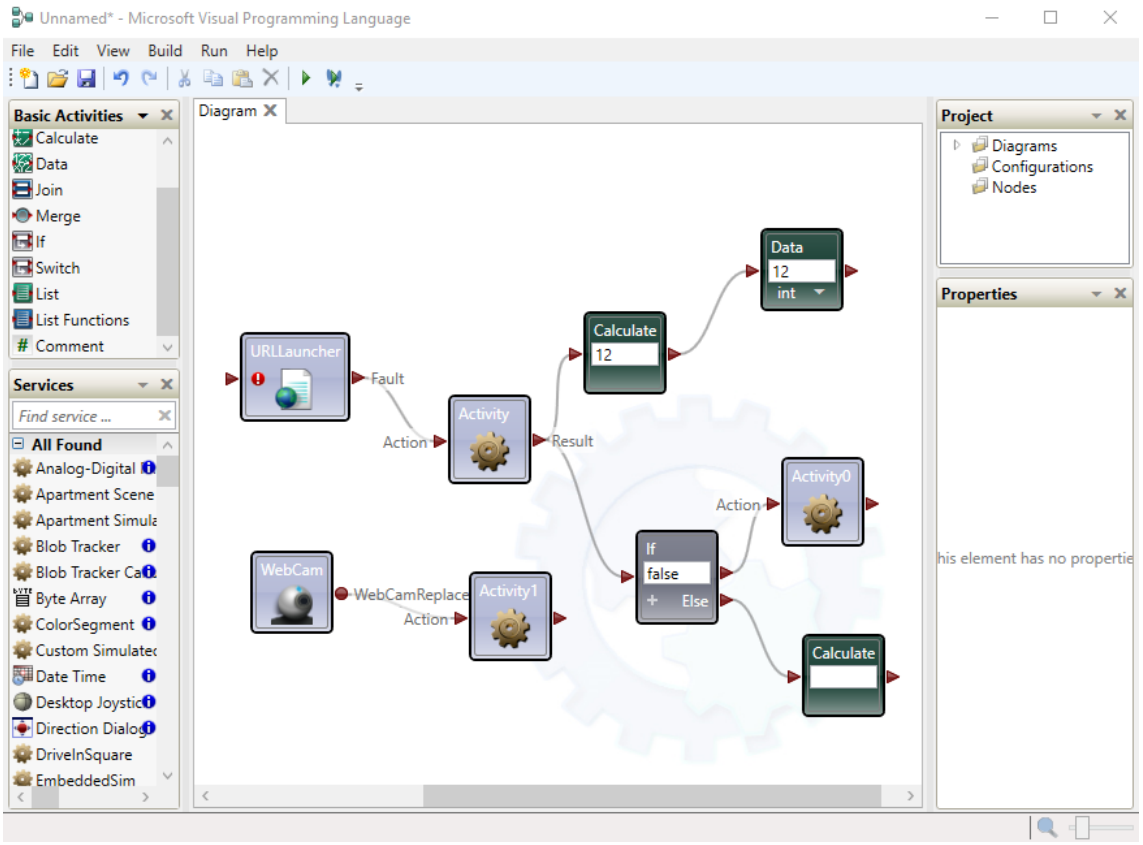
Eelnevad näited lõppkasutajale lihtsa ja arusaadava programmeerimiskeele või -platvormi loomiseks on tekstipõhised ning seetõttu mitte-programmeerijate jaoks keerulised õppida ja kasutada. Seetõttu on välja arendatud visuaalse programmeerimise kontseptsioon, milles kasutatakse graafilisi elemente ja tekstiline programmikood on lõppkasutaja eest varjatud. [5] [6]

Kõige levinum paradigma visuaalse programmeerimise puhul on *jigsaw* ehk puslemeetod [5]. Selle puhul kasutab programmeerija eelkirjeldatud, pusletükke meenutavaid elemente, mida saab omavahel kindlate reeglite alusel ühendada. Kokkusobivad elemendid tehakse kasutajale arusaadavaks sarnaselt pusletükkide ühendamisele – kui kaks elementi omavahel kokku ei sobi siis on nende ühendusliidesed erineva kujuga.



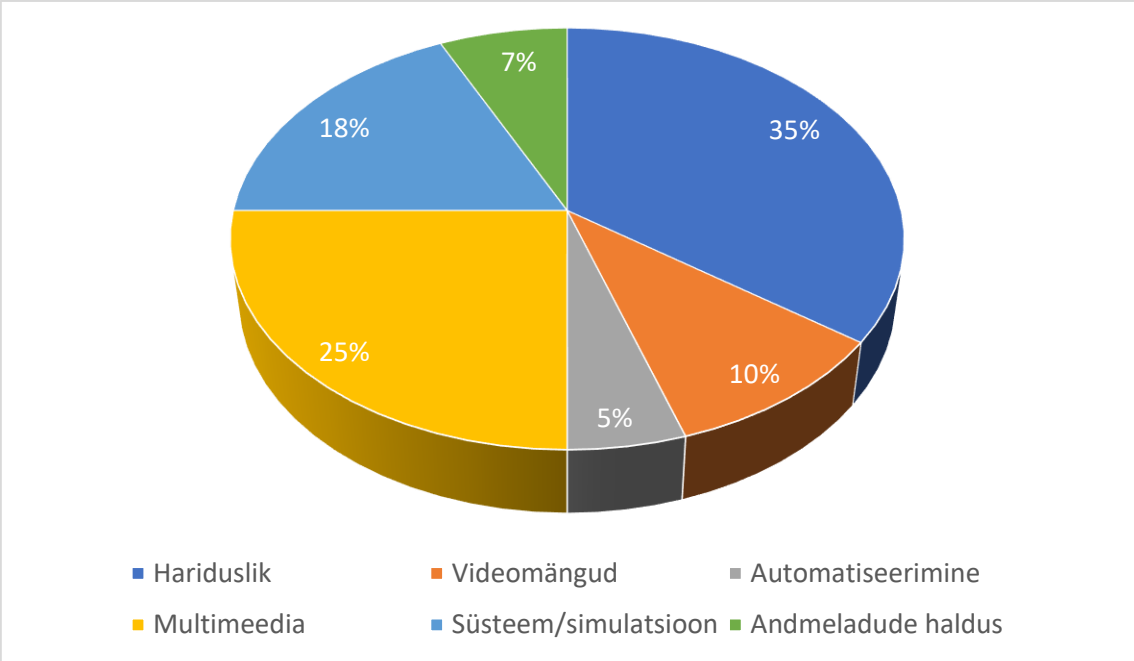
Joonis 1. Scratch programmeerimiskeskkonna ekraanipilt. (Allikas: autori koostatud)

Teine stiil, mida kasutatakse on vooskeemid (*flow-diagrams, flowchart*) [5] [6]. Erinevalt puslemeetodist ühendatakse vooskeemide puhul erinevad elemendid joontega ning see lisab rohkem paindlikkust. Vooskeeme kasutatakse näiteks Microsofti VPL (*Visual programming language*) [7] tootes, mida kasutatakse robootikas robotite jaoks koodi kirjutamiseks. B# nimeline programmeerimiskeel on loodud vooskeemide põhiselt [8]. On olemas ka puslemeetodi ja vooskeemide hübriide (Lego EV3 Programmer APP) [5].



Joonis 2. Microsoft VPL ekraanipilt. (Allikas: autori koostatud)

Erinevate visuaalsete programmeerimiskeelte jaotus 2017.a. seisuga oli järgnev [9]:



Joonis 3. Visuaalsete programmeerimiskeelte jaotus valdkondade lõikes [9].

Visuaalse programmeerimise valdkonnas ei ole välja kujunenud üldiseid programmeerimiskeeli nagu on tekstipõhiste keelte puhul (Java, C++, Javascript jne.).

See on tingitud sellest, et visuaalse programmeerimisega lahendatakse konkreetset probleemi, näiteks [9] [10]:

- Lastele programmeerimise aluste õpetamine
- Robotite juhtimime ilma koodi tundmata
- Asjade interneti (*IoT*) programmeerimine
- Lõppkasutaja programmeerimise võimaldamine olukordades, kus koodi kirjutamine ei ole mõistlik

Kuna eelnevas nimekirjas väljatoodud kasutusjuhtude eesmärgid ja sihtgrupid on äärmiselt erinevad on üldise visuaalse programmeerimiskeele loomine suur väljakutse. Hetkel ei ole olemas ühtset mudelit, mida saaks igas olukorras kasutada ning seetõttu luuakse üldiselt iga rakenduse jaoks oma spetsiifiline keel. [9]

Selliseid spetsiifilisi „keeli“ nimetatakse domeenipõhisteks keelteks (ingl kl. *domain-specific languages*). Domeenipõhised keeled on samuti täisväärtuslikud, kuid loodud konkreetse tarkvara või plat–vormi jaoks. Sellisel juhul on kasutusel ka spetsiifilised terminid, keel toetab täpsemini kasutaja vajadusi sellel platvormil ning on optimeeritud sellel platvormil või rakenduses kasutamiseks. [10]

Domeenipõhise keele loomisel on kolm aluspõhimõtet [10]:

1. Peavad olema kaardistatud korrektsed nõuded
2. Tuleb koostada sobiv arhitektuurne lahendus
3. Tuleb luua tööriistad, et lõppkasutaja saaks kirjutada selles domeenipõhises keeles

Domeenipõhise keele loomine sisaldab järgnevaid faase [11]:

1. Otsustamine kas luua domeenipõhine keel või mitte
2. Analüüs
3. Disain
4. Realisatsioon
5. Kasutuselevõtmine

Käesolevas magistritöös käsitletav platvorm, mis võimaldab selle kasutajatel lihtsasti luua enda tingimustele vastavaid spetsiifilisi pilvepõhiseid rakendusi nõuab domeenipõhise keele loomist. Autor eeldab seda järgnevatel põhjustel:

1. Sellise platvormi kasutaja ei soovi selgeks õppida mõnda üldist programmeerimiskeelt kuna tema tegevusvaldkond ja eriala ei ole seotud programmeerimisega.
2. Ka lihtsamate rakenduste loomiseks kulub tekstipõhiste programmeerimiskeeltega märkimisväärselt aega. Graafilise liidesega platvorm võimaldaks arendamisele kuluvat aega tunduvalt lühendada.
3. Puudub üldotstarbeline programmeerimiskeel, mida saaks kasutada sellise platvormi loomiseks.

3.4 Visuaalse programmeerimise eelised LKP osas

Thomas B. Hilburn, kes õpetas ülikoolis arvutiteaduste aineid, leidis et traditsioonilised viisid programmeerimise algõpetamiseks ei ole tudengite jaoks piisavalt motiveerivad ning tekitavad tihti segadust. Põhjus on selles, et programmeerimise õpetamist alustatakse keele süntaksist ja lihtsamatest ülesannetest samal ajal, kui programmeerimine kujutab endast tegelikkuses hoopis probleemide kaardistamist ja nende lahendamist. Selleni jõutakse aga alles õppekavade viimastel kursustel. Taolist traditsioonilist lähenemist nimetas ta „alt-üles“ meetodiks. [12]

Et muuta tarkvaraarendust tudengite jaoks selgemaks ja lihtsamini õpitavaks töötas ta välja meetodi, mida nimetas „ülevalt-alla“ meetodiks. Selle meetodi puhul õpetati esmalt üldiseid tarkvara arendamise põhimõtteid ja protsesse, probleemist arusaamist ja selle lahenduse kirjeldamist. Lahenduse implementatsioon tuli kõige viimasena, kui hakati õppima konkreetsemalt programmeerimiskeelt ja selle rakendamist. Seega sai tudengite jaoks programmeerimiskeelest tööriist olemasolevate probleemide lahendamiseks, mitte lihtsalt vahend kursuse läbimiseks. Selle meetodi tulemusel tõusis tudengite motivatsioon ja nad omandasid järgnevates ainetes teadmisi kiiremini, kui „alt-üles“ meetodiga õppinud tudengid. [12] [8]

Visuaalsed programeerimiskeskkonnad toetavad üldiselt seda teooriat, sest need vähendavad rutiinseid tegevusi, mis kaasnevad tekstilise programmikoodi kirjutamisega – näiteks sulgude korrektne kasutamine, rea lõpetamine semikooloniga jne. Seega saab tarkvara kirjutav inimene suunata oma tähelepanu pigem tarkvara loogikale ja probleemide lahendamisele, kui konkreetse keelesüntaksi jälgmisele. [8]

Inimese paremat arusaama graafiliselt kujutatud objektidest kinnitab ka D. Scanlani poolt läbi viidud psühholoogia alane uuring. Ta õpetas ülikoolis andmestruktuure ning pani tähele, et õpilased saavad teooriast tunduvalt paremini aru, kui andmestruktuurid ja programmi loogika on pseudokoodi asemel kujutatud graafiliselt vooskeem (*flow-chart*). Täpsemad uuringud näitasid, et 75% tudengitest said pigem aru graafilistest joonistest, kui teksti kujul koodist. [8] [13]

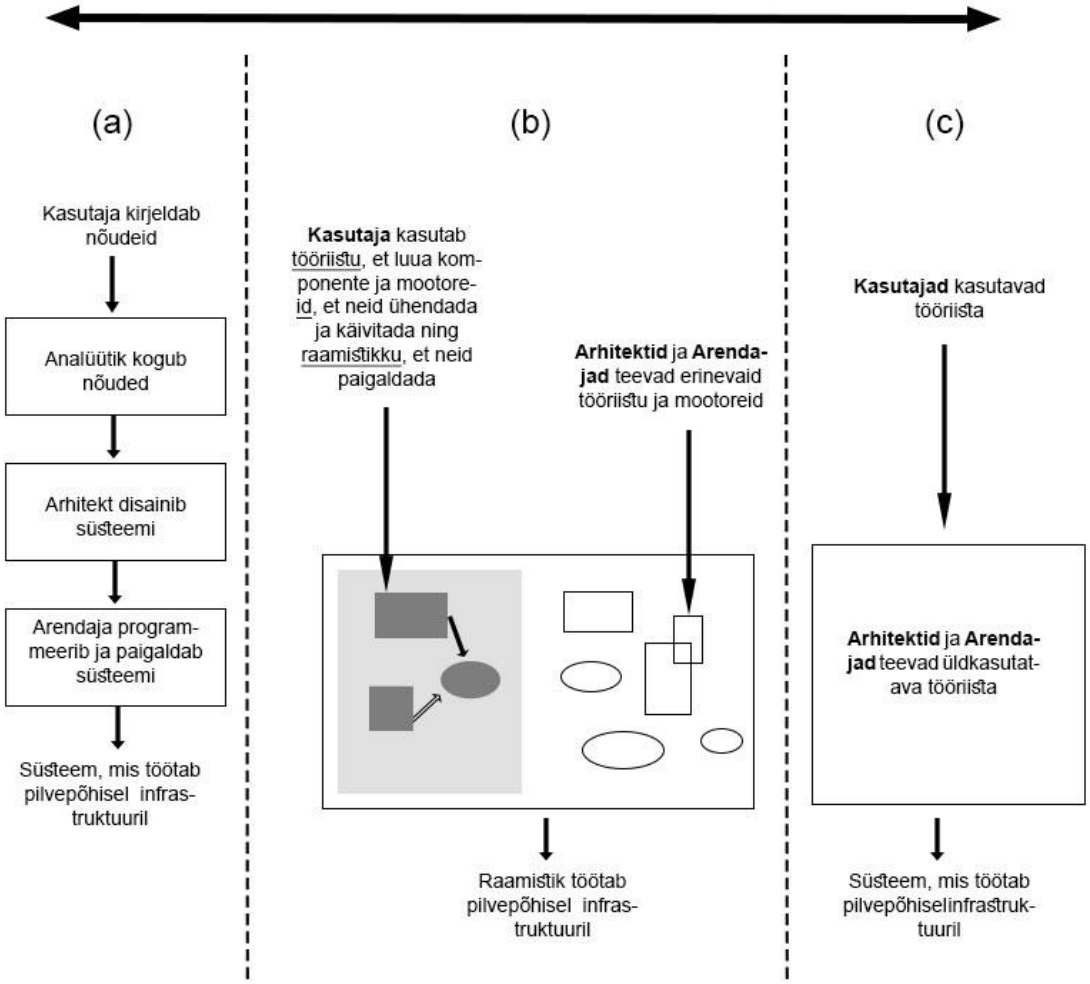
Guangtong, Qiliang ja Jianchun on välja toonud visuaalse programmeerimise kolm suuremat eelist võrreldes tekstipõhise programmeerimisega [14]:

1. Kognitiivsus – kuna inimesed tajuvad objekte pigem visuaalselt siis on sel viisil tunduvalt lihtsam infot omandada ja sellest aru saada. Näiteks suudavad erinevat keelt rääkivad inimesed omavahel suhelda palju edukamalt käte abil, kui teha üksteisele selgeks konkreetsete sõnade tähendus teises keeles.
2. Keele omadused – tekstil on ainult üks dimensioon ehk tähtede ja sümbolite jada. Ühel pildil on aga korraga mitmeid dimensioone – näiteks kuju, värv, suurus, muster, suund ja kaugus. Seega on võimalik ühte objekti kirjeldada pildina tunduvalt lihtsam, kui tekstina. Siia võib veel juurde märkida eelnevast punktist, et sama pilt on arusaadav erinevat keelt kõnelevatele inimestele.
3. Kasutusmugavus – visuaalsetel programmeerimiskeeltel on tavaliselt kindel kasutusmuster, kõrge efektiivsus (vähe vigu), kasutajasõbralikkus, väike õppimiskõver jne. Kasutajal on vaja vaid teada, mida tema tarkvara lõpuks tegema peab ning vastavad moodulid kokku tõstma. See on eriti sobiv algajatele ja mitte-elukutselistele programmeerijatele.

Visuaalse programmeerimise eeliseks ei ole ainult programmeerimise lihtsustamine inimeste jaoks, kes ei oska ühtegi tekstilist programmeerimiskeelt. Selle eelis võib väljenduda ka selles, et visuaalsete komponentide abil tarkvara kokkupanek võib olla tunduvalt kiirem, kui seda teksti kujul kirjutades (jälgides samal ajal vastava keele süntaksireegleid). Graafiliste komponentide abil programmeerimise kohta on tehtud mitmeid uurimusi. Nendes uuringutes on kasutajatele välja pakutud graafilisi tööriistu tarkvara nõuete ja struktuuri kujutamiseks. Selle põhjal saab kasutaja kirjeldada detailsemad spetsifikatsioonid ja komponentide ning andmete visualiseerimine aitab kaasa keerulisemate rakenduste loomisele. [15]

3.5 Lahendused visuaalse programmeerimiskeskkonna loomiseks

Tarkvaralahenduse tellijate poolelt vaadates on levinud kaks meetodit (vt. Joonis 4). Esimene (vt. joonisel „a“) meetod on traditsiooniline tarkvaraarenduse tellimine, mille negatiivsed küljed on välja toodud peatükis 3.2. Põhiliselt tekivad selle meetodi puhul probleemid sellest, et tellija ei oska enda nõuded piisavalt detailselt kirjeldada ja arendaja ei oska ette näha tulevikus tekkida võivaid nõudeid, millega peaks juba alguses arvestama. Teine variant kaldub teise äärmusesse (vt. joonisel „c“), kus tellija teeb kogu äriloogikat puudutava arenduse ise. Selle eelduseks on WYSIWYG-tüüpi tööriistad, mis võimaldavad programmeerimist mitteoskaval inimesel seda teha. Sellisteks tööriistadeks on näiteks FrontPage ja Dreamweaver aga ka tänapäeval laiemalt levinud veebilehtede arendamise platvormid wix.com, voog.ee jne. Selle meetodi miinuseks on aga see, et need tööriistad võimaldavad küll valmis teha staatilisi veebilehti, kuid keerulisemaid veebirakendusi, mis nõuavad andmete sisestamist, töötlemist ja loogikat need tööriistad ei võimalda. [3, p. 3]



Joonis 4. Võimalikud lähenemised veebirakenduste arendamisele. [3, p. 4]

Kuna mõlemal eelnevalt nimetatud variandil oli palju piiranguid on välja pakutud vahepealne variant (joonisel „(b)“), kus tarkvaraarendajad kirjutavad komponente, mida tellijad saavad kasutada ning omavahel siduda luues niimoodi endale sobiliku tarkvaralahenduse. See meetod toetab ka inkrementaalset tarkvaraarendust ehk kogu tööriistade ja komponentide komplekt ei pea kohe olemas olema vaid neid saab vastavalt vajadusele juurde lisada. [3, p. 3]

J. Ginige, De Silva ja A. Ginige [3] toovad välja, et lõppkasutaja programmeerimise kontseptsioon koos komponentide põhise arhitektuuri ja inkrementaalse arendusmetoodikaga aitavad lahendada probleeme, mis esinevad väikeettevõtetel tarkvara tellimisel. Kuna need tööriistad ja komponendid on taaskasutatavad ja programmeerimisega tegeleb ettevõtte enda töötaja on võimalik kokku hoida kuludelt, püsida graafikus ja vajalikke muudatusi kiiremini sisse viia [3, p. 4].

Lizcano ja teised [2] on välja toonud faktorid, mida lõppkasutaja programmeerimise platvormi mudel peaks sisaldama:

1. Iga komponent peaks olema lõppkasutaja jaoks „must kast“ ehk täitma ühte konkreetset ülesannet. See lihtsustab kasutajal komponentide mõistmist ja nende kasutamist probleemide lahendamisel. Lisaks peab komponent andma visuaalselt edasi seda, mida see teeb. Kasutaja peaks selle funktsioonist aru saama ning mõistma mida see komponent teeb süvenemata kuidas see komponent tegelikult sisemiselt enda ülesannet täidab.
2. Kasutajal peab olema võimalik määrata andmevoogusid erinevate komponentide vahel. Mitte-programmeerija jaoks peab olema selline seadistamine loogiline ja vältima programmeerimiskeelte süntaksit. Selleks tuleb luua abstraktsed mudelid. Soovitatav on kasutada lihtsaid näidisandmeid ja komponentide ühendusi (andmete edastamist) võimalikult loogiliselt visualiseerida, et need oleksid kasutajale mõistetavad. Lisaks peab olema võimalik täpsustada andmete sisu, et see oleks hiljem mõistetav ka teistele kasutajatele, kes soovivad samu komponente kasutada.
3. Kasutajad peavad saama andmevoogusid kirjeldada ajaliselt ja ruumiliselt. Neil peab olema näiteks võimalus kuvada kasutajaliideses erinevaid tulemusi vastavalt konkreetsetele andmetele, protsessidele jne.
4. Kuna kõikidel kasutajatel ei ole teadmisi komponentarhitektuurist peavad platvormi poolt pakutavad komponendid olema valmislahendused ehk mitte-programmeerija ei peaks hakkama ise komponente looma. Olenevalt rakenduse eesmärgist võib lisada ka keerulisemaid komponente, millede abil saab kasutaja ise kokku panna mõne vajaliku komponendi. Sellised elemendid tuleb aga kindlasti eristada, et need ei tekitaks segadust kasutajates, kes ei ole piisavalt teadlikud ja soovivad kasutada lihtsamaid valmislahendusi. Üks võimalus on luua komponentide hierarhia, kus kõrgemal asetsevad nö. valmiskomponendid ning hierarhias sügavamal on elemendid, mida saab kasutada enda komponentide loomiseks.

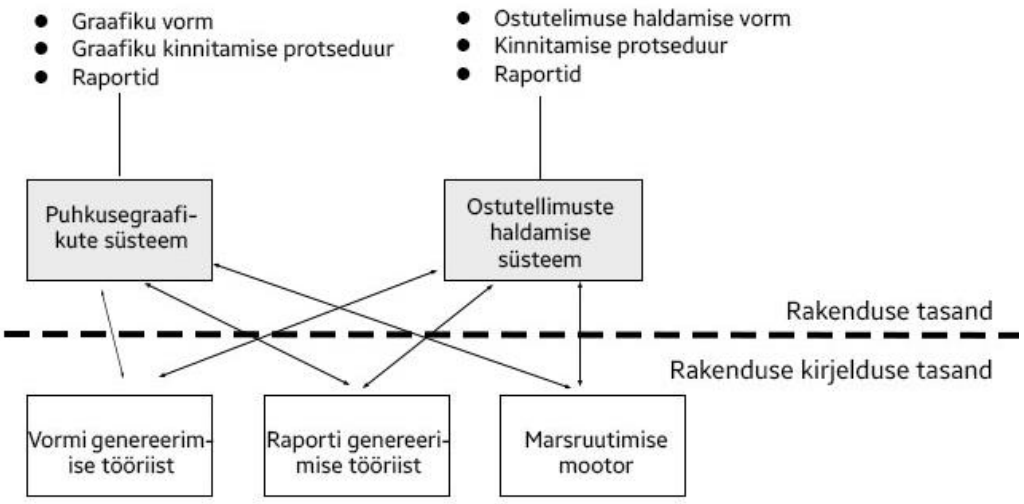
J. Ginige, De Silva ja A. Ginige [3] võtsid aluseks eelnevad uuringud, mis käsitlesid veebirakenduste kõige olulisemaid aspekte ja erinevaid veebirakenduste tüüpe. Need elemendid ja tüübid pandi ühte tabelisse ning tehti uus uuring, mille käigus paluti

kogenud veebirakenduste programmeerijatel tuua välja, kui tähtsad on konkreetsed elemendid erinevat tüüpi rakendustes. Tulemused toob välja Tabel 1.

Tabelis on välja toodud erinevad rakenduse osad. Need osad tuleb lahendada kas raamistiku või komponentidena. Otsus sõltub sellest, kui oluline ning mida täpselt konkreetne osa hõlmab. Näiteks peab raamistik toetama nõudeid, mis on ühised kõigile veebirakenduse tüüpidele, samas saab komponentidega lahendada rakenduse tüübi spetsiifilisi olukordi [3].

Raamistikus toetatava funktsionaalsuse näitena võib tuua veebirakenduse struktureerimise ja navigatsiooni võimaluse. Komponendina saab näiteks lahendada sisestatud andmete valideerimise. Selliselt on võimalik kasutaja eest peita liigseid komponente ja vähendada õppimise keerukust. Kasutaja saab selgeks õppida põhilise raamistiku ning kasutada spetsiifilisi komponente vaid vajadusel [3].

Joonis 5 kujutab näidet, mille puhul on samade komponentide taaskasutamisel loodud kaks erinevat rakendust. Töötajate puhkusegraafikute ning ostutellimuste haldamise tarkvarad kasutavad samu komponente – vormide ja raportite genereerimise tööriistu ning navigatsiooni haldamise vahendit [3, p. 5].



Joonis 5. Samade komponentidega loodud erinevad rakendused. [3]

Tabel 1. Veebirakenduse elementide tähtsus erinevat tüüpi rakendustes [3].

Rakenduse tüüp / **Olulised aspektid veebirakendustes**	**Informatiivne**	**Otsing, kausta- ja sõnastikuotsingu rakendused**	**E-poed ja veebiportaalid**	**Meelelahutus, suhtlemine ja sõnumite edastamine**	**Koostööd ja projektijuhtimist võimaldavad platvormid**	**Teenustele orienteeritud või kõrgtaseme veebiarenduse tööriistad**
Veebisaidi struktureerimine ja navigatsiooni käsitlemine	keskmine-kõrge	keskmine	kõrge	kõrge	kõrge	kõrge
Lehe mallid ja vormide loomine	keskmine	keskmine	keskmine-kõrge	keskmine-kõrge	kõrge	keskmine-kõrge
Sisestatud andmete valideerimine	madal-keskmine	madal-keskmine	keskmine-kõrge	madal-keskmine	kõrge	kõrge
Vormi andmete sidumine andmebaasi/tabeliga	madal-keskmine	madal-keskmine	keskmine-kõrge	keskmine-kõrge	kõrge	kõrge
Kasutajate autentimine ja ligipääsukontroll	madal-keskmine	madal	keskmine-kõrge	madal-keskmine	kõrge	kõrge
Seansi haldamine	madal-keskmine	madal	keskmine-kõrge	madal-keskmine	keskmine-kõrge	keskmine-kõrge
Ärireeglite esindatus rakendustes	madal	madal	keskmine-kõrge	madal	kõrge	keskmine
Andmebaasipäringud ja tingimuslikud väljundid	keskmine-kõrge	kõrge	keskmine-kõrge	keskmine-kõrge	keskmine-kõrge	kõrge

Rakenduse tüüp / Olulised aspektid veebirakendustes	**Informatiivne**	**Otsing, kausta- ja sõnastikuotsingu rakendused**	**E-poed ja veebiportaalid**	**Meelelahutus, suhtlemine ja sõnumite edastamine**	**Koostööd ja projektijuhtimist võimaldavad platvormid**	**Teenustele orienteeritud või kõrgtaseme veebiarenduse tööriistad**
Andmebaaside loomine ja muu sellega seonduv (andmemudelid, normaliseerimine, indekseerimine jne.)	madal-keskmine	kõrge	keskmine-kõrge	keskmine-kõrge	keskmine-kõrge	keskmine-kõrge
Andmete turvalisus nende salvestamisel ja edastamisel	keskmine-kõrge	madal-keskmine	kõrge	madal	keskmine-kõrge	keskmine
Multimeedia võimalused (videode striimimine jne.)	madal-keskmine	madal-keskmine	madal-keskmine	kõrge	madal-keskmine	keskmine

4 Analüüs

Järgnevalt viiakse läbi analüüs loodava platvormi nõuete ja reeglite kaardistamiseks. Esmalt uuritakse olemasolevaid tooteid ja koostatakse esmane prototüüp. Lõppkasutajatelt tagasiside saamiseks koostatakse küsimustik, mille käigus tutvustatakse ka prototüüpi ja küsitakse selle kohta tagasisidet. Lisaks analüüsitakse platvormi tehnilisi nõudeid ja piiranguid ning pakutakse välja sobivad lahendused.

4.1 Olemasolevate lahenduste võrdlus

Analüüsitava rakenduse analooge on turul vähe. Kaks levinuimat on „tadabase“ [16] ja „Bubble“ [17]. Tadabase on uuem platvorm, käesoleva töö kirjutamise ajal on see veel *beta* versioonis ja seetõttu kõikidele kasutajatele tasuta. Ekraanipilt tadabase'i kasutajaliidesest on välja toodud käesoleva töö lisas 5. Platvorm võimaldab selle kasutajatel luua andmebaasitabeleid, erinevaid lehti ning siduda loodud tabelite andmeid lehtedega. Sealjuures on kogu rakendus võimalik valmis ehitada ilma koodi kirjutamata ning see ongi platvormi loojate eesmärk. Loodud rakendust hoiustatakse Tadabase'i poolt seega ei ole vaja kasutajal otsida eraldi hostinguteenuse pakkujat.

Tadabase'i positiivsed omadused:

1. Võimalik määrata kasutajaid ja seeläbi ka rakendada tarkvara erinevatele osadele kasutajapõhiseid ligipääsupiiranguid.
2. Koodivaba programmeerimine ehk veebirakenduse saab edukalt valmis teha inimene, kes ei oska ühtegi programmeerimiskeelt ja ei oma teadmiseid veebirakenduste traditsioonilisest arendusest.
3. Valminud rakendusele saab määrata enda domeeninime ehk lõppkasutaja ei saa eru, et veebirakendus on loodud selle tööriistaga.
4. Veebirakendus skaleerub erineva suurusega ekraanide jaoks.

Tadabase'i negatiivsed omadused:

1. Platvormi kasutajaliidese stiil põhineb Bootstrap [18] raamistikul ja see seab väga tugevad piirangud. Kuna kasutatakse olemasolevaid komponente siis see lihtsustab platvormi arendamist, kuid sunnib ka selle kasutajaid samasid printsiipe jälgima. Näiteks koosnevad kõik lehed ridadest ja need omakorda veergudest. Kasutaja ei saa määrata elementidele suvalist asukohta.
2. Tabelite genereerimine ei ole intuitiivne. Tabelid ja andmed lisatakse vormide abil, mille puhul ei ole kasutajal alati selge ülevaade mida ta hetkel teeb.
3. Platvorm võimaldab ka andmete pärimist erinevatest tabelitest korraga, kuid selle kasutamine tundub algaja jaoks keeruline.
4. Vaikimisi on määratud vaid üks stiil, mille puhul on menüüriba ekraani üleval ääres. Menüüriba asukohta muuta ei saa, lisaks ei saa ise määrata menüüvalikuid ega nende loogikat.
5. Keerulisemate modifikatsioonide jaoks eeldatakse kasutajalt CSS'i ja Javascripti tundmist.
6. Valminud rakendused ei näe disaini poolest kuigi head välja.

Bubble on kauem turul olnud ning 2019.a. aprilli seisuga on kodulehe andmetel nende tootel 259 500 kasutajat. Kuna see toode on juba algsest arendusjärgust väljas siis on selle kasutamine üldjuhul ka tasuline. Hinnad algavad 14 dollarist (12,45 eurot) kuus ning kui on soov liita arendajate hulka rohkem kasutajaid võib kuutasu tõusta kuni 445 dollarini (395,73 eurot) kuus. Ekraanipilt Bubble'i kasutajaliidesest on leitav käesoleva töö lisast 6.

Funktsionaalsuse osas on Bubble kindlasti Tadabasest võimsam. Kasutajaliides on arusaadavam ning kindlasti rohkem läbimõeldud. Selle platvormi eesmärk on samuti võimaldada *drag and drop* funktsionaalsuse abil luua endale sobivaid veebilehti ning programmeerida kasutajaliides selliselt, et andmed oleksid seotud andmebaasitabelitega ning elementidele saaks lisada tingimusi. Näiteks saab kasutaja määrata mis juhtub, kui klõpsatakse nupule ja millistel tingimustel teatud tegevusi tehakse.

Bubble'i positiivsed omadused:

1. Väga võimalusterohke. Suurem osa vajalikest elementidest on olemas.
2. Lehtede koostamine on mugav, elemente saab lohistada sobivasse kohta ja disainida vastavalt vajadusele.

3. Programmeerimise lihtsustamiseks saab luua enda kujundusstiile, et neid hiljem mugavalt elementide lisamisel kasutada.
4. On loodud palju pluginaid, et anda kasutajatele lisavõimalusi. Näiteks liidestused erinevate API-dega.
5. On olemas tõlkimisvõimalus.
6. Põhjalik dokumentatsioon ja viisard esmakordsel kasutamisel. Lisaks on koostatud interaktiivsed lühikursused tarkvara õppimiseks.

Bubble'i negatiivsed omadused:

1. Päris algaja jaoks on valikuvõimalusi liiga palju. Kui elemendid on võimalik lihtsasti lehele lohistada siis nende sidumine andmete ja tingimustega ei ole enam kuigi intuitiivne.
2. Mitmed kasulikud pluginad on tasulised ja ei sisaldu kuutasus.

Kokkuvõttes on mõlemal platvormil positiivseid omadusi, millega peab kindlasti uue platvormi loomisel arvestama. Näiteks kasutajaõiguste määramine, veebilehtede loomine visuaalse programmeerimise abil ja enda domeeniga sidumine. Väga oluline on kindlasti ka tõlkimisvõimalus ja lihtne liidestamine erinevate API-dega. Samas on autori arvates need platvormid päris algaja tarkvara looja jaoks kohati liiga keerulised ning kasutajaliidest saab kindlasti intuitiivsemaks teha. Loodav rakendus ei tohiks toetuda ühele CSS-raamistikule selliselt, et ka kogu disain (k.a. värvid) on sellelt üle võetud ning kasutajad ei saa ise muudatusi teha.

4.2 Esmane prototüüp

Järgnevalt on kirjeldatud esmast prototüüpi projekteeritavale platvormile. Prototüübi koostamine on eeldus ka järgmisele etapile ehk kasutajatelt tagasiside kogumisele. Prototüüpe koostatakse erinevates arendusettapides. Kui on juba olemas nõuded saab neid prototüüpimisel aluseks võtta, kui nõudeid ei ole luuakse prototüüp oletatavatel nõuetel või eeldustel. Viimast varianti kasutatakse juhul, kui luuakse uut tarkvara ning ei ole veel tehtud kasutajauuringuid või püstitatud süsteeminõudeid. Prototüübiga kontrollitakse eelduseid ning selle tulemusel märgitakse eeldus otse nõudeks, muudetakse seda ning seejärel kirjeldatakse nõudena või jäetakse üldse kõrvale. Kuna käesoleva töö eesmärk on

luua uus süsteem ja puuduvad eelnevad nõuded siis luuakse prototüüp eeldustel. [19, p. 32]

Järgnevalt on välja toodud eeldused esmase prototüübi koostamiseks:

1. Tegemist on veebirakendusega
2. Rakendusel on kaks vaadet – redigeerija rakenduste loomiseks ja lõppkasutaja vaade loodud rakenduse kuvamiseks ja kasutamiseks.
3. Rakenduses saab luua erinevaid vaateid.
4. Prototüübis kasutatakse andmetabeli, kasutajate ligipääsu, navigatsiooni ja lehe sisu genereerimise tööriistu.
5. Kuna tegemist on kasutajatele esmase funktsionaalsuse tutvustamisega siis prototüüp ei sisalda kõiki veebirakenduse elemente.
6. Esmane prototüüp ei võimalda kasutada programmeerimiskeeltest tuntud tingimuslikke plokke – näiteks *if/else* tingimused.

Prototüübi loomisel kasutati veebipõhist töövahendit Figma [20], mis võimaldab lihtsasti luua interaktiivseid ja klikitavaid prototüüpe. Selle tulemusel on võimalik lõppkasutajale pakkuda võimalikult lõpptulemuse lähedast kogemust. Osa prototüübist loodi ka reaalse veebirakendusena ehk programmeeriti kasutades HTML, CSS ja Javascripti programmeerimiskeeli. Selle tingis vajadus kuvada kasutajatele visuaalse programmeerimise ühte aspekti ehk elementide lohistamist ekraanil enda õigele positsioonile, mida Figmaga ei oleks olnud võimalik realiseerida.

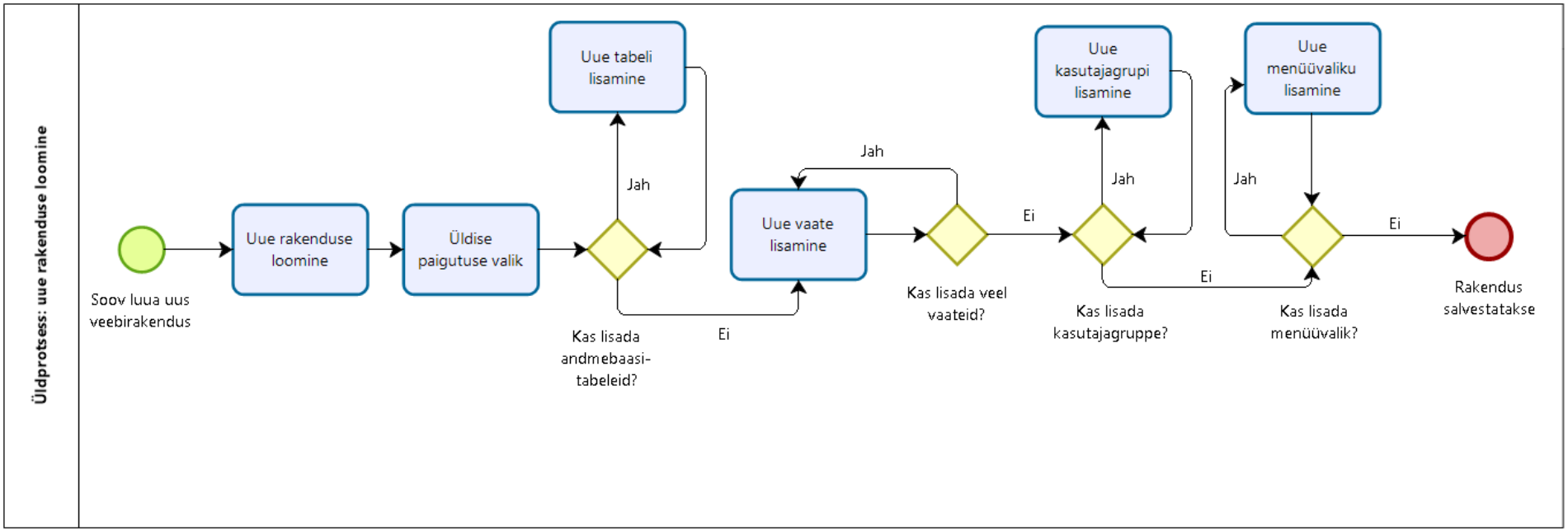
Loodud prototüübi ekraanipildid on välja toodud lisas 1. Nendel on näha ühe veebipõhise näidisrakenduse loomine, mille kasutusjuht on müügiosakonna töötajatele uute klientide andmebaasi sisestamine. Prototüübist tehti video, mis tutvustab töövoogu ja erinevaid funktsioone. Video eesmärgiks on analüüsitava rakenduse lõppkasutajale tutvustamine tagasiside kogumiseks.

Esmase prototüübi töövoog on järgnev:

1. Kasutaja loob uue rakenduse
2. Valitakse rakenduse paigutus ehk menüüriba asukoht
3. Lisatakse (vajadusel) uued andmebaasitabelid. Tabelites saab sisestada veergude pealkirjad ning valida veeru andmetüübi

4. Lisatakse põhivaated ehk lehed, mille vahel on hiljem võimalik navigeerida. Vaate loomisel kasutakse *drag and drop* ehk lohistamise meetodit. See tähendab, et kasutaja saab lehel kõik elemendid hiirega sobivasse asukohta lohistada. Iga element on teatud osas muudetav – näiteks värv, raamid, tekstisuurus.
5. Lisatakse (vajadusel) kasutajagrupid, et hiljem oleks võimalik rakendada erinevatele lehtedele ja menüüvalikutele ligipääsupiiranguid.
6. Lisatakse (vajadusel) menüüvalikud, et lõppkasutajal oleks mugavam navigeerida. Menüüvalikud saab siduda konkreetse lehega ja vajadusel kasutajagrupiga. Vaikimisi on menüüvalik avalik.
7. Loodud rakendus salvestatakse.

Joonis 6 kirjeldab esmase prototüübi töövoogu ühe uue rakenduse loomisel.



Joonis 6. Esmase prototüübi töövoog (Allikas: autori koostatud)

4.3 Metoodika valik kasutajate tagasiside kogumiseks

Käesolev magistritöö käsitleb uue infosüsteemi loomist. Kuna see infosüsteem on mõeldud kasutajatele, kellel puuduvad programmeerimisalased oskused ja tõenäoliselt ei ole soovi seda ka õppida tuleb neil võimaldada kasutada visuaalset programmeerimist. Kasutajaliidese poolele tuleb seega luua domeenipõhine keel (vt. p.t. 3.2).

Selle infosüsteemi kasutajad ei ole piiritletud konkreetse organisatsiooni ega tegevusvaldkonnaga. Seetõttu ei saa selle platvormi arendamisel võtta aluseks ühe konkreetse inimgrupi oskuseid, soove ega demograafilist jaotust.

Eelnevat arvestades peab kasutajaliideses olev visuaalse programmeerimise keskkond vastama selle platvormi kasutajate oskustele ja vajadustele. Seega tuleb domeenipõhise keele loomiseks läbi viia kasutajauuringud, et välja selgitada mida täpsemalt kasutajad selliselt platvormilt ootavad ning leida vastused järgnevatele püstitatud küsimustele (vt. p.t. 2):

- Kui lihtne ja intuitiivne peab kasutajaliides olema, et tavakasutaja suudaks selle eelneva koolituse või õppeta kasutusele võtta?
- Kui palju on keskmine kasutaja nõus panustama aega uue tarkvara kasutama õppimisele?
- Kas on üldse mõistlik kasutada visuaalset programmeerimist?

Kasutajatelt tagasiside kogumiseks on kaks peamist meetodit – kvalitatiivsed- ja kvantitatiivsed uuringud. Kvalitatiivsed tehnikad (näiteks fookusgrupid ja intervjuud) annavad vastused küsimusele *miks* kasutajad teevad mingeid teatud tegevusi aga ei anna ülevaadet millised konkreetsed karakteristikud seda tarkvara kasutavad. Ainult kvantitatiivsed uuringud (näiteks küsitlused, küsimustikud) saavad anda vastuse näiteks küsimustele millised vanusegrupid uuritavat tarkvara kasutavad ja kui suur hulk kasutajatest sooviks näha planeeritavaid uuendusi. Kvantitatiivsete uuringute tulemusel on võimalik piiritleda kasutajad, kelle peal viia läbi kvalitatiivseid uuringuid ja määratleda mida täpsemalt nende käigus kasutajatelt küsida. [21]

Et saada ülevaadet käesoleva töö käigus projekteeritava infosüsteemi kasutajatest ja püstitada esmased nõuded tuleb läbi viia küsitlus. Võimalikult laiapõhjalise tulemuse

saamiseks viiakse küsitlus läbi küsimustiku vormis. Järgnevalt kirjeldatakse selle küsimustiku koostamist ja ülesehitust.

4.4 Küsimustik nõuete kaardistamiseks

Küsimustik koosneb 24 küsimusest ning on jagatud kolme plokki. Iseloomustavad küsimused võimaldavad vastajaid grupeerida vanuse, asukoha, haridustaseme ning ameti ja tegevusvaldkonna järgi. See on oluline saamaks infot selle kohta kas näiteks mõni tegevusvaldkond oleks antud platvormi jaoks välistatud või kas mõni vanusegrupp sooviks harjumuste tõttu jääda kasutama pigem ettevõttes olemasolevaid tarkvarapakette.

Käitumuslikele aspektidele suunatud küsimused toovad välja praeguse olukorra – kui levinud on ettevõtetes tabeltöötlustarkvarad ning kui palju nendes hoiustatavatest andmetest sõltub. Lisaks uuritakse, kui suure tõenäosusega võib tekkida tabelites vigu ja kuidas neid maandatakse.

Suhtumisega seotud küsimused on suunatud inimeste rahulolule olemasolevate tarkvarapakettidega ning kas nad sooviksid võimalusel neid täiustada või muuta. Lisaks uuritakse, millised on tarkvara juures kõige olulisemad ja millised vähemolulised näitajad. Näiteks tarkvara kaasaegsus, kasutamise lihtsus, õppimise kiirus jne. Pärast üldisemaid küsimusi kuvatakse vastajale esmase prototüübi videot ning järgnevad küsimused on seotud konkreetselt sellega. Sealhulgas kas vastaja oleks valmis sellist platvormi kasutama ning kas see lahendaks temal endal või tema ettevõttes mõne konkreetse probleemi.

Siinkohal on tähtis ära märkida, et antud küsimustik ei lahenda konkreetseid aspekte platvormi loomisel – näiteks kas elementide seadistuse valikud peaksid asuma kohe elemendi kõrval või peaksid need olema ekraanil eraldi plokis. Kuna tegemist on täiesti uue platvormi loomise esimese etapiga siis on kasutajauuring suunatud pigem üldiste harjumuste ja trendide väljaselgitamiseks. Lisaks tekib antud uuringuga esmane teadmine ja tagasiside kas planeeritava platvormi järele on üldse vajadus olemas.

Platvormi arendamise protsessi käigus tuleb kindlasti läbi viia täiendavaid uuringuid, mis on pigem kvalitatiivsed ehk näiteks intervjuud. Need uuringud peaksid andma täpsemad vastused kasutusmugavuse ja kasutajakogemuse osas.

Järgnevalt on välja toodud küsimustik:

Iseloomustavad küsimused

1. Vanus
2. Asukoht (riik)
3. Haridustase
4. Ametikoht (spetsialist, keskastmejuht, tippjuht, töötu, ettevõtja)
5. Tegevusvaldkond

Käitumuslikud küsimused

6. Kas kasutad tööalaselt näiteks Google Spreadsheet, Excel vms tabelarvutustarkvara?
7. Kas neid tabeleid jagatakse ettevõttesiseselt andmete jagamiseks?
8. Kas neid tabeleid täidavad erinevad inimesed?
9. Kui suur tähtsus on nendes tabelites olevatel andmetel teie ettevõtte jaoks?
10. Kas eksisteerib risk, et keegi kustutab kogemata andmed, täidab tabeleid valesti jms?
11. Kui jah siis kuidas maandate seda riski?
12. Kas ettevõtte andmete (ladu, töötajad, klienditugi, suhtlus, dokumendid) hoiustamiseks ja haldamiseks kasutate ka mõnda muud tarkvara?

Suhtumisega seotud küsimused

13. Kas oled/olete rahul tabeltöötlustarkvarade võimalustega? (ligipääs, õigused, jagamine)
14. Kas oled/olete rahul muude kasutuselolevate tarkvarade võimalustega? (ligipääs, õigused, jagamine)
15. Kui olulised on järgnevad võimalused sinu jaoks:
    a. Tarkvara peab olema kaasaegne
    b. Turvalisus
    c. Muudatused kooskõlas kasutajate soovidega
    d. Kasutajatugi
    e. Võimalikult palju modifitseerida
    f. Kasutamise lihtsus

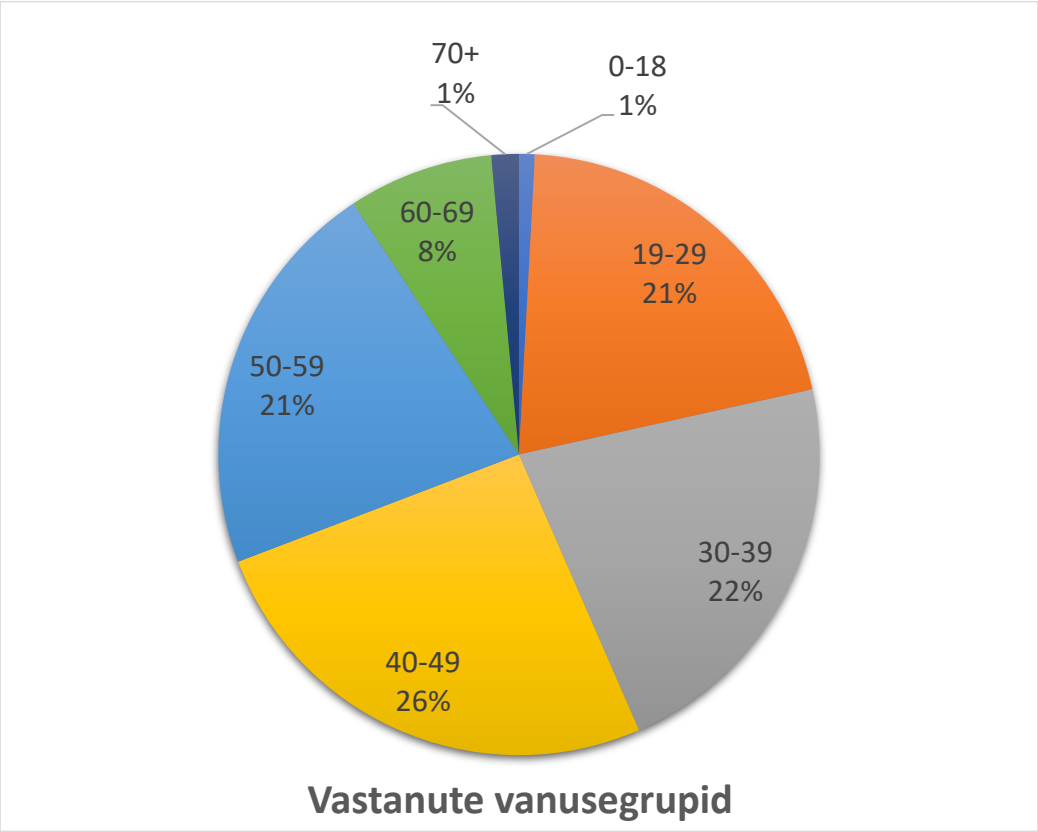
g. Õppimise kiirus

16. Kui saaksid vabalt muuta kasutuselolevaid tarkvarapakette ja lisada neile enda jaoks sobivaid võimalusi siis kui tõenäoliselt seda teeksid?
17. Kui sul oleks võimalik ise koostada enda või ettevõtte vajadustest lähtuv veebirakendus, mis asendaks olemasolevad Exceli/Google Spreadsheet tabelid/failid siis kas oleksid valmis neist loobuma?
18. Kui palju oleksid nõus enda aega panustama endale sobiva tarkvara koostamisele eeldusel, et see annab sulle/teile võimalused, mida praeguses tarkvaras ei ole.
19. *Kuvatakse esmast prototüüpi videona*
20. Kui tõenäoliselt kasutaksid sellist tööriista endale individuaalse tarkvara loomiseks?
21. Kas see tööriist lahendaks sinu või sinu ettevõtte jaoks olulisi probleeme, mis tekivad seoses kasutuselolevate tarkvaradega?
22. Kas kasutaksid selliselt loodud tarkvara pigem ettevõttesiseselt või looksid veebirakenduse ka klientidele?
23. Kas sellise tarkvara puhul on ka oluline kasutajaõiguste ja ligipääsude seadistamise võimalus?
24. Kas oleksite nõus osalema intervjuus täpsema tagasiside kogumiseks?

4.5 Kasutajauuringu tulemused ja analüüs

Kuna loodava platvormi potentsiaalsed kasutajad ei ole kuidagi piiratud siis saadeti küsimustik välja ilma sihtgruppi filtreerimata. Seega võis küsitluse päringu saada igas vanuses ning igas valdkonnas tegelev inimene. Suurem osa küsitlusel osalemise päringutest saadeti e-posti teel eesti ettevõtete avalikele e-posti aadressidele. Selliseid päringuid saadeti umbes 9500. Et saada vastuseid ka väljastpoolt Eestit, tehti vastavaid postitusi ka reddit.com ning Amazon Mechanical Turk keskkondades. Küsitluses osalemise pöördumises kirjeldati lühidalt käesoleva magistritöö eesmärki ning vastamise korral pakuti võimalust olla platvormi esimeste testkasutajate hulgas.

Küsitlusele vastanuid oli kokku 474 ja seda kõikidest vanusegruppidest (vt. Joonis 7). 19-29 ja 50-59 aastaseid oli vastanute hulgas peaaegu võrdselt, teistest veidi enam esines 40-49 aastaseid vastanuid. Ülejäänud vanusegruppidest oli vastanuid vähem.

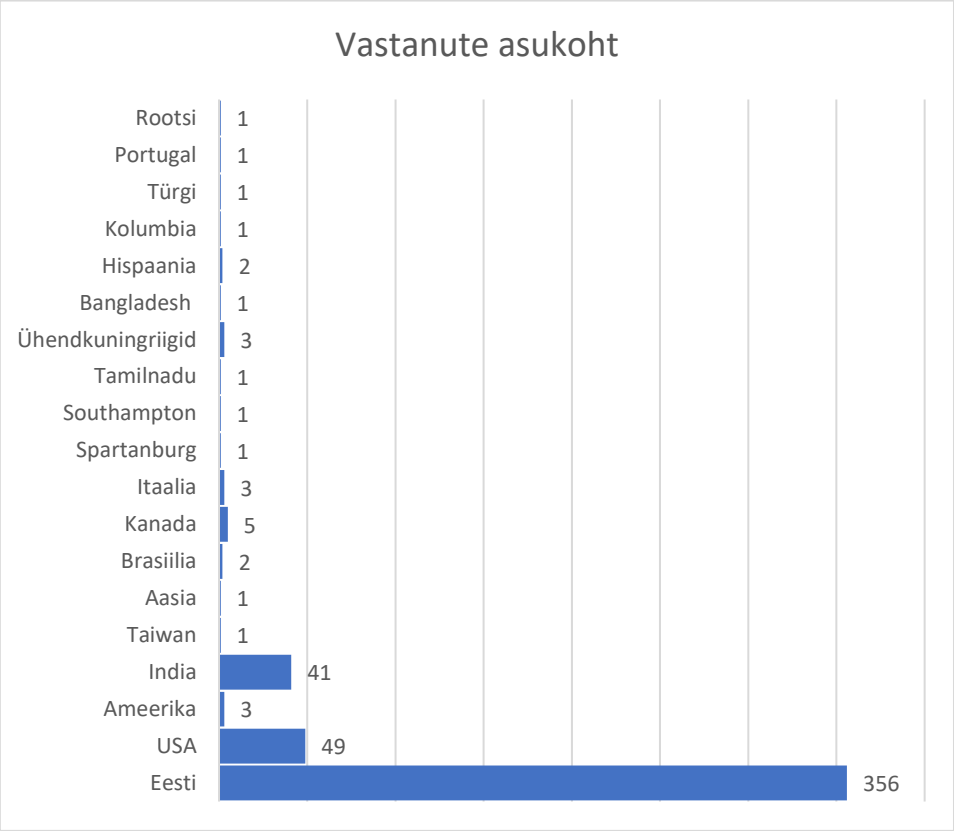


Joonis 7 Küsitlusele vastanud vanusegruppide lõikes (Allikas: autori koostatud)

Erinevaid asukohti oli märgitud 19. Ankeedis oli küll palutud märkida riik, kuid mõned üksikud vastajad olid märkinud kas linna või kontinendi täpsusega (vt. Joonis 8). Kuna kõige suurem osa küsitluse päringutest saadeti Eesti ettevõtetele siis on oodatult ka suurim vastajate osakaal Eestist, kokku 356 vastanut. Järgnesid USA (49 vastanut) ja India (41 vastanut).

Haridustaseme alusel jagunesid vastanud järgmiselt:

- Kõrgharidus/bakalaureusekraad – 40%
- Magistrikraad – 33%
- Kutsekeskharidus – 12%
- Keskharidus – 10%
- Doktorikraad – 3%
- Muu – 2%
- Põhiharidus – 0,4%



Joonis 8. Küsitlusele vastanud asukoha lõikes (Allikas: autori koostatud)

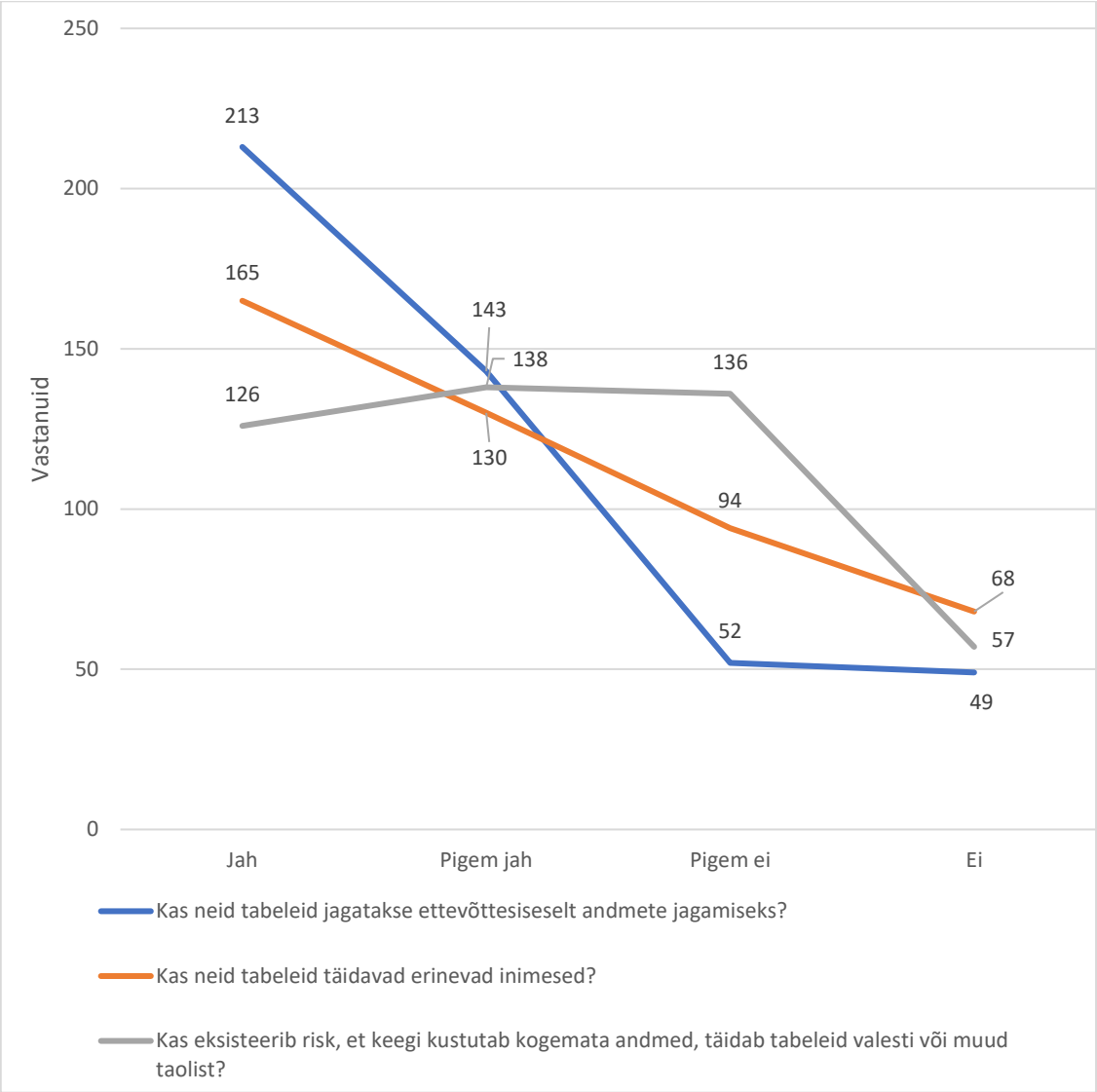
Ametikoha järgi oli jagunemine järgnev:

- Ettevõtja – 32%
- Spetsialist – 25%
- Keskastmejuht – 21%
- Tippjuht – 10%
- Muu – 8%
- Töötu – 3%
- Lihttööline – 1%

Seega oli 85% vastanutest vähemalt kutsekeskharidusega, mille põhjal võib järeldada, et on läbitud vähemalt minimaalne arvutikasutamise õpe. Ka küsitluse saatmise viis filtreeris juba välja need inimesed, kes arvutit üldse ei kasuta. Positiivne tulemus oli see, et vastanute hulgas oli palju ettevõtjaid. Ettevõtte suurust küll ei küsitud, kuid kuna suurem osa vastajatest asusid Eestis siis võib järeldada, et pigem on tegemist mikro- või

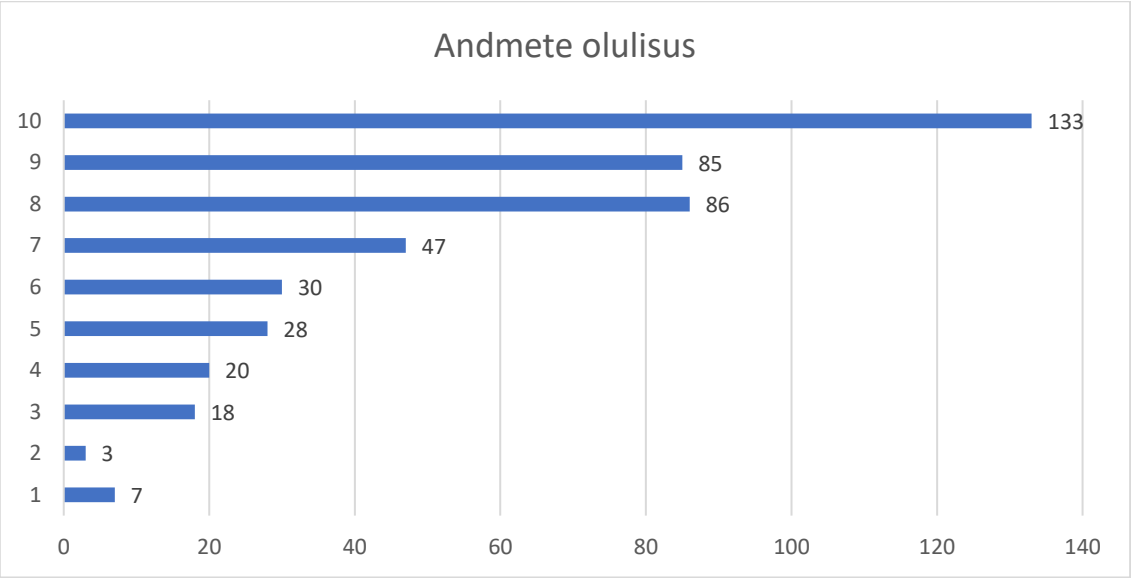
väikeettevõtetega, millele ongi loodav platvorm suunatud. Erinevaid tegevusvaldkondi märgiti 264.

Küsimusele kas kasutatakse tööalaselt tabelarvutustarkvara vastas jaatavalt 92%. Järgnevalt uuriti, kas koostatud tabeleid jagatakse ka ettevõttesiseselt info jagamiseks ja kas neid tabeleid täidavad erinevad isikud. Lisaks küsiti hinnangut riskile, kui keegi täidab tabelit valesti (näiteks sisestab andmed valesse lahtrisse), kustutab andmeid või muud taolist (vt. Joonis 9).78% vastanutest vastas, et jah või pigem jah jagatakse tabeleid erinevate isikute vahel. 64% puhul täidavad samu tabeleid erinevad inimesed. Nende tulemuste vahe seisneb tõenäoliselt selles, et jagatavad tabelid ei ole alati mõeldud teistele täitmiseks vaid teatud juhtudel jagatakse vaid nendes sisalduvat infot. Üpris suur osa, 58% vastanutest, hindas tõenäoliseks riski, et keegi muudab või kustutab ekslikult tabelites infot, mida tegelikult muuta või kustutada ei tohiks. Arvestades seda, et inimesed hindavad ise riske väiksemaks, toetab see uuringutes välja toodud tulemusi, et 86-100% tabeltöötlustarkvarades koostatud tabeleid sisaldavad vigu (vt pt. 3.2).



Joonis 9. Tabelite kasutusviisid ettevõtetes (Allikas: autori koostatud)

Tabelites sisalduvate andmete olulisust ettevõtte jaoks hinnati pigem kõrgeks, 48% vastanutest märkis 10-pallisel skaalal andmete olulisuseks 9 või 10 (vt. Joonis 10). Arvestades seda, et vastajad ise hindasid andmete kadumise või muutumise riski kõrgeks ning tegemist on ettevõtte jaoks oluliste andmetega võib järeldada, et tegemist on olulise probleemiga andmete haldamise osas.



Joonis 10. Andmete olulisus jagatavates tabelites (Allikas: autori koostatud)

Peamised lahendused, mis selle probleemi osas välja toodi on järgmised:

- Failide koopiad, versioonikontroll
- Ristkontroll, kontrollarvud ja valemid
- Tabelitega tegelevad ainult konkreetsed isikud
- Märgitakse tabelis, mida muuta ei tohi
- Loodetakse, et keegi ei tee vigu
- Andmete kontroll

Palju toodi välja seda, et neid riske ei maandatagi või aktsepteeritakse seda riski. Üllatavalt palju vastuseid tõid lahenduseks selle, et lihtsalt usaldatakse tabeli täitjat ja probleemide korral küsitakse täitjalt üle. Mitmed vastajad märkisid, et probleemide vältimiseks täidabki andmeid vaid üks isik. Samuti lisati, et see suurendab selle isiku töökoormust ning vastutust. Suurem osa riskide maandamise viise sisaldasid siiski inimfaktorit ehk keegi kontrollib andmed üle või teeb failidest käsitsi koopiad. Toodi välja ka automaatsete koopiate lahendusi ja versioonikontrolli, kuid võrreldes käsitööga oli selle osakaal siiski väiksem. Nende vastuste põhjal võib järeldada, et andmete säilimise ja vigade vältimise süsteem on väga oluline. Lisaks peab kindlasti olemas olema versioonikontroll, et vajadusel oleks võimalik andmeid siiski taastada.

57% vastajatest märkis, et nende ettevõttes kasutatakse andmete hoiustamiseks ja haldamiseks lisaks tabeltöötlustarkvaradele ka muud tarkvara. Kõige rohkem toodi välja

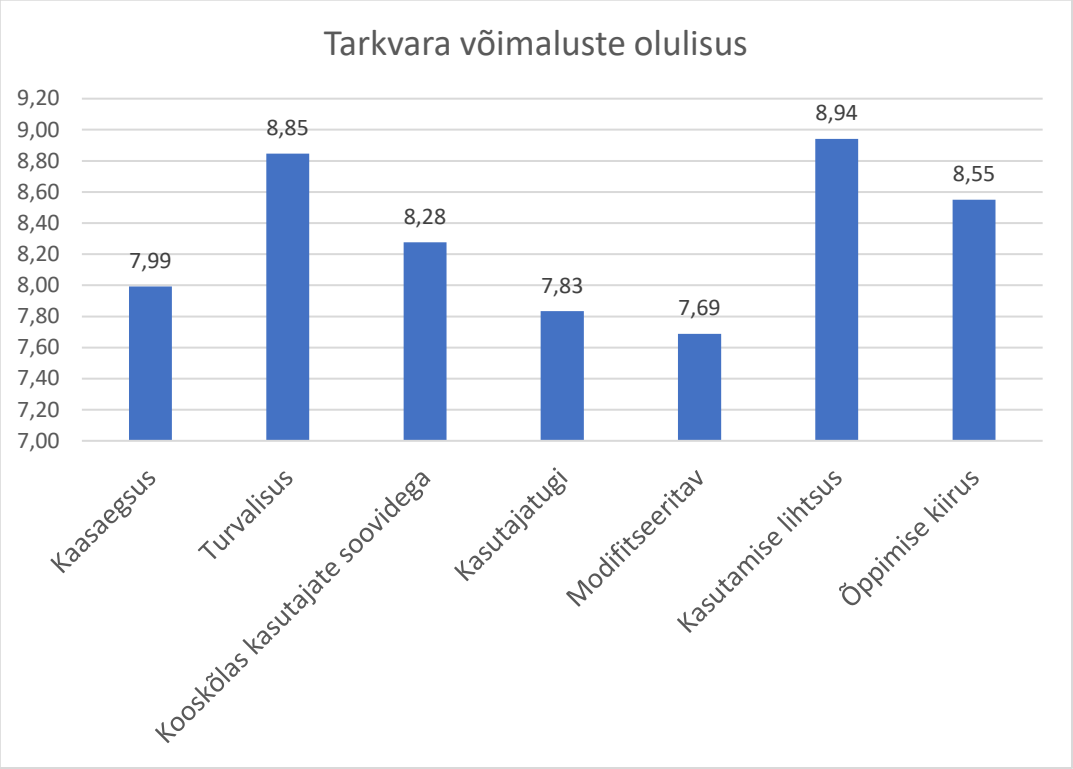
erinevaid raamatupidamistarkvarasid ja valdkonnaspetsiifilisi tarkvarasid. Samas olid üpris levinud ka Google Drive, Dropbox ja Microsoft One Drive lahendused. Mõned üksikud ettevõtted kasutavad ka enda loodud või spetsiaalselt tellitud tarkvara.

90% vastajatest on rahul tabeltöötlustarkvarade võimalustega ning 88% on rahul muude tarkvarade võimalustega (ligipääs, õigused, jagamine). Samas paluti ka täpsustada, millega vastajad nende tarkvarade puhul rahul ei ole. Põhilised probleemid olid järgmised:

- Kasutajate õiguseid ei saa ise muuta
- Tarkvara versioonid muutuvad tihti ja erinevad versioonid ei ole omavahel ühilduvad
- Samaaegselt saab tarkvara või faili kasutada vaid üks inimene
- Kasutusmugavus on halb
- Kasutajaliidese muutumine uute versioonidega
- Ei sisalda kõiki vajalikke mooduleid või mõned vajalikud valdkonnad on katmata
- Liiga palju peab tegema manuaalset tööd
- Faile peab jagama e-maili teel
- Lisakasutajate lisamine on tasuline
- Mitme kasutajaga samaaegne kasutamine on andmete muutmise osas riskantne või ei ole tarkvara sel juhul alati töökindel
- Ei ole piisavalt oskuseid ja teadmisi tarkvara kõiki võimalusi kasutada
- Andmebaasi ei saa hoiustada pilveteenuses
- Tarkvara ei vasta kõigile nõudmistele
- Uuendustega muutuvad seadistused
- MTÜ ei saa lubada endale kallist, kuid kvaliteetset tarkvara. Seetõttu kasutatakse alternatiive, mis ei vasta täielikult nõuetele.
- Ei saa kasutada erinevatel platvormidel või vähemalt kuvada andmeid erinevatel seadmetel
- Turvalisus on kaheldav, kui tekib vajadus andmeid teistele jagada

Peaaegu kõik nimetatud probleemid on seotud tarkvaraga, mitte välise mõjutajaga, näiteks internetiühenduse kiirus, lõppkasutaja arvuti võimsus vms. Seega on need probleemid tarkvara arendamise käigus lahendatavad. Välja toodi ka neid probleeme,

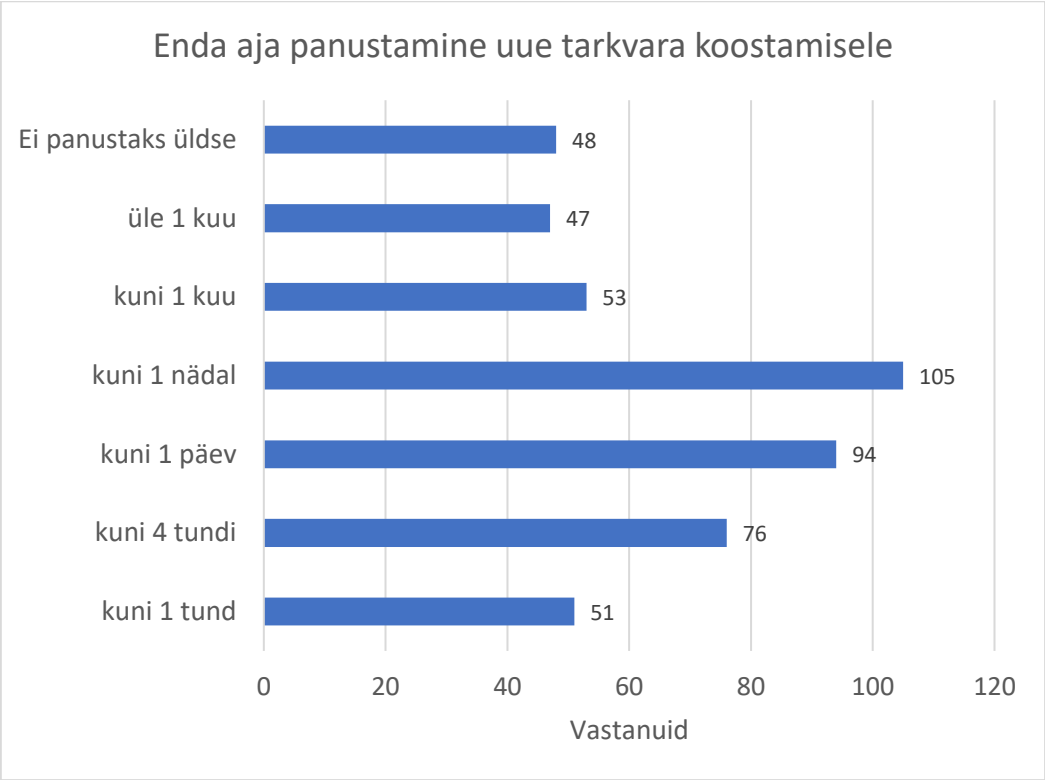
mida antud töö käigus analüüsitava platvormiga soovitakse lahendada – näiteks andmete hoiustamine pilveteenuses, samaaegne kasutamine, kasutajaõiguste määramine (vt. pt. 2).



Joonis 11. Tarkvara poolt pakutavate võimaluste tähtsus vastajate jaoks 10-pallisel skaalal (Allikas: autori koostatud)

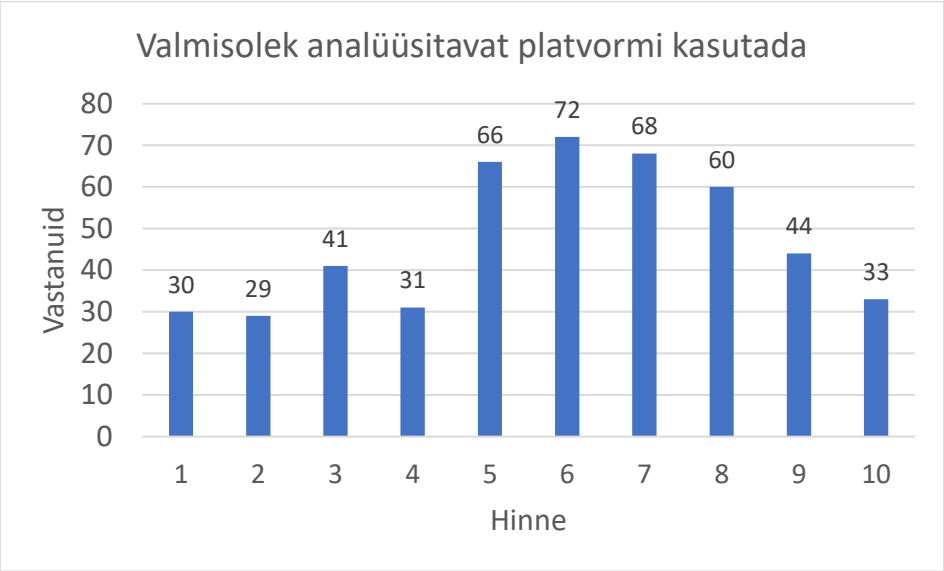
Küsimusele „Kui saaksid vabalt muuta kasutuselolevaid tarkvarapakette ja lisada neile enda jaoks sobivaid võimalusi siis kui tõenäoliselt seda teeksid“ anti 10-pallisel skaalal keskmiseks hindeks 7,2. 55% kasutajatest andsid sellele hindeks 8-10. Olemasolevate tabelarvutustarkvarade loobumise keskmiseks hindeks anti 6,72 ja kõrgema hinde ehk 8-10 andis 47% kasutajatest. Siinkohal määrab ilmselt olulist rolli ka küsimuse sõnastus kuna analüüsitav platvorm ei ole mõeldud asendamaks tabeltöötlustarkvarasid kuna nende võimalused spetsiifilistel kasutusjuhtudel on väga head.

Küsitlusele vastanud on nõus ootamatult palju aega panustama endale sobiva tarkvara koostamisele (vt. Joonis 12). 63% vastanutest on valmis panustama 5 tundi kuni üle 1 kuu. Enda aega ei oleks nõus üldse panustama 10% vastanutest. See tulemus näitab, et vastajate soov selliste võimaluste järgi, mida kasutuselolevad tarkvarapaketid ei võimalda, on piisavalt suur, et ollakse valmis sellele märkimisväärselt enda aega panustama.



Joonis 12. Vastajate valmisolek panustada enda aega uue tarkvara koostamisele (Allikas: autori koostatud).

Järgnevalt kuvati vastajatele analüüsitava platvormi esmase prototüübi (vt. pt. 4.2) videot ning paluti hinnata, kui tõenäoliselt nad oleksid valmis seda kasutama endale individuaalse tarkvara loomiseks (vt. Joonis 13). Keskmine hinne oli 5,86. Küsimusele „Kas see tööriist lahendaks sinu või sinu ettevõtte jaoks olulisi probleeme, mis tekivad seoses kasutuselolevate tarkvaradega“ vastas „Jah“ või „Pigem jah“ 42% vastanutest ehk 201 inimest. Sealjuures oli ametikohtade lõikes hinnang erinev, kõige suurema tõenäosusega (6,49 punkti) hindasid selle kasutamist keskastmejuhid ning järgmisena tippjuhid (6,20 punkti). Reeglina ongi ettevõtetes nendel ametikohtadel olevate inimeste eesmärk tööprotsesse automatiseerida ning saada algandmetest otsuste tegemiseks vajaminevaid korrektseid aruandeid.



Joonis 13. Küsitlusele vastajate valmisolek kasutada analüüsitavat platvormi. Põhineb näidatud prototüübi videol. (Allikas: autori koostatud)

Sellise platvormiga loodud tarkvara kasutaks ainult ettevõttesiseselt 36% kasutajatest, klientide jaoks kasutaks seda 24%. 40% ei osanud selles küsimuses seisukohta võtta. 89% vastanutest leidis, et sellise platvormi puhul on oluline kasutajaõiguste ja ligipääsude seadistamise võimalus. 474 vastanust soovis olla esimeste testkasutajate hulgas 175, kes jätsid selleks ka enda e-posti aadressi.

4.6 Funktsionaalsed nõuded

Funktsionaalsed nõuded kirjeldavad, kuidas süsteem peaks käituma kasutajapoolsete või teisest süsteemist pärinevate sisendite peale ehk millised on need tegevused ja teenused, mida süsteem peab pakkuma.

Analüüsitava süsteemi funktsionaalsed nõuded kujunevad autori poolt püstitatud ning kasutajauuringu põhjal tekkinud nõuetest. Kuna agiilses tarkvaraarenduses on nõuded pidevas muutumises siis käesolevas töös nimetatud nõuded on platvormi arendamise esimese etapi sisenditeks.

Kuna analüüsitav platvorm võimaldab luua veebirakendusi siis on järgnevalt selgitatud mõisted, mida funktsionaalsete- ja mittefunktsionaalsete nõuete kajastamisel kasutatakse.

Platvorm – Analüüsitav veebipõhine rakendus, mis võimaldab mitte-programmeerijatel luua veebirakendusi.

Veebirakendus – Tarkvara, mis on loodud **platvormi** kasutaja ehk kliendi poolt.

Kasutaja – **Platvormi** klient (mitte-programmeerija), kes loob endale või teistele **platvormi** abil **veebirakendusi**.

Lõppkasutaja – Inimene, kes kasutab loodud **veebirakendust**.

4.6.1 Veebirakenduse loomise mooduli nõuded

Veebirakenduse loomise moodul on **platvormi** osa, kus klient ehk mitte-programmeerija saab luua veebirakenduse, millel on kasutajaliides ning andmebaas ja omab kõiki põhilisi veebirakenduse elemente.

1. **Kasutaja** saab luua veebitarkvara visuaalse programmeerimise abil ehk ei pea **veebirakenduse** loomiseks kasutama tekstipõhist programmeerimiskeelt.
2. **Kasutaja** saab vabalt kirjeldada sobival kujul andmebaasitabeleid
3. Andmebaasitabeli loomisel peab **kasutaja** veeru lisamisel valima ka andmetüübi. Andmetüübid kuvatakse nimekirjana ning tavakasutajale arusaadavalt (näiteks „DATE()“ on „Kuupäev“, „INT“ on „Arv“ jne.).
4. **Kasutaja** saab ise määrata andmebaasitabeli nime ning veergude nimed.
5. **Kasutaja** saab valida **veebirakenduse** üldise paigutuse ehk kus asub ekraanil peamenüü. Võimalikud menüü asukohad on üleval, paremal, vasakul, all. Menüü täidab valitud asukoha täies laiuses või kõrguses.
6. **Kasutaja** saab loodud tabelit täita ka käsitsi, sarnaselt tabeltöötlustarkvarale.
7. **Kasutaja** saab luua **veebirakenduses** erinevaid vaateid ehk lehti.
8. **Platvorm** pakub kasutajale valmis HTML-komponente, mida on võimalik hiirega ekraanil sobivasse kohta lohistada. Asukoht ei ole piiratud. Asukohta saab ka hiljem muuta.
9. Ekraanile paigutatud elementide atribuute on võimalik kasutajal muuta. Vastavalt elemendile näiteks suurus, värv, vari, nurkade kumerus jne.
10. Soovi korral saab ekraanile paigutatud elemente kustutada.
11. **Kasutaja** peab saama andmebaasitabelites olevaid andmeid siduda HTML-elementidega loodaval vaatel. Selleks võib luua spetsiaalsed komponendid

(tabelid, vormid, väljad, graafikud), et mitte-programmeerijal oleks lihtsam tabeleid ja elemente omavahel seostada (vt. ptk. 3.5 p.2 ja p.4).

12. **Veebirakendus** peab võimaldama andmeid andmebaasitabelisse lisada. Selleks peab kasutaja saama lisada vastavaid vorme.
13. Andmebaasitabelisse andmete lisamise komponent ei tohi nõuda kõikide andmebaasiväljade täitmist (kui väli ei ole tabelis märgitud kohustuslikuks). **Kasutaja** peab saama valida, millised andmebaasiväljad konkreetse vormiga täidetakse.
14. Kasutajaliidese komponent peab võimaldama CRUD-toiminguid – andmete lisamist, lugemist, uuendamist ja kustutamist andmebaasist.
15. Elementide hulgas peab olema valik faili(de) üleslaadimiseks. Komponent peab võimaldama lisada failile identifikaatorit (kausta nimi), mille alusel **kasutaja** saab hiljem faile filtreerida.
16. Teksti või pildielement peab pakkuma võimalust viidata üleslaetud failile.
17. Platvormil peab olema element üleslaetud failide nimekirja kuvamiseks. Element peab võimaldama failide kuvamiseks nende filtreerimist (kausta tasemel).
18. Loodav **veebirakendus** peab vaikimisi toetama kasutajate ligipääsupiiranguid.
19. **Veebirakenduse** seadete vormil peab **kasutaja** saama valida kas autentimine on nõutav. See tähendab, et **kasutaja** soovib kogu rakenduse sisu teha privaatseks ja avalehel kuvatakse alati sisselogimise vormi.
20. **Kasutaja** peab saama lisada erinevaid kasutajagruppe ning gruppidesse **lõppkasutajaid**. Esimeses versioonis on ligipääs lahendatud e-posti aadressi alusel.
21. **Lõppkasutaja** lisamisel peab saama sisestada vähemalt tema nime (ei ole nõutav) ja e-posti aadressi (nõutav).
22. Kasutajagrupi vaates kuvatakse selles grupis olevate **lõppkasutajate** nimekiri.
23. Uue **lõppkasutaja** lisamisel saadetakse e-posti aadressile ajalise piiranguga veebiaadress, mis avab vormi, kus **lõppkasutaja** saab endale parooli määrata.
24. **Kasutaja** peab saama eraldi vormil lisada menüüvalikuid ning määrata milline vorm (leht) konkreetse menüüvaliku alt avaneb. Menüüvalikule saab määrata kasutajagrupi, mis seda menüüvalikut näeb. Täpsem **lõppkasutajate** ligipääsude määramine lisandub järgmiste versioonidega.

25. **Kasutaja** peab saama näha graafikutel rakenduse tehnilisi detaile, näiteks võrguliiklust teatud perioodi jooksul, kasutatud ja järelejäänud andmemahtu, andmekeskuse asukohta.
26. **Kasutaja** peab saama muuta rakenduse põhiseadistusi. Näiteks andmete salvestusmaht, andmekeskuse asukoht, rakenduse nimi, rakenduse domeeninimi.

4.6.3. Administraatori mooduli nõuded

Administraatori moodul on **platvormi** haldaja (omaniku) vaade, mis võimaldab saada ülevaadet klientidest, koostada neile arveid, jälgida arveldussaldosid, saata teavitusi jne.

1. **Platvormil** peab olema nimekiri kõikidest klientidest ja nende andmetest. Nii klientide nimekirja, kui ka üksiku kliendi detailvaates.
2. **Platvormil** peab olema aruanne klientide valitud pakettidest ning võimalus neid seadistada ja muuta näiteks hinda (kokkulepe, hinnaalandus vmt.).
3. Klientide nimekiri peab olema filtreeritav nime, asukoha, paketi ja muude andmete alusel, et lihtsustada teavituste saatmist.
4. (Filtreeritud) klientidele peab olema võimalik saata HTML-vormingus e-kirju.
5. Iga kliendi detailvaates peab olema võimalik näha tema loodud **veebirakendusi** ja nende tehnilisi andmeid – võrguliikluse koormust, kasutatud andmemahtu.
6. Liiga kõrge võrguliikluse koormusega või andmemahu täituvusega **veebirakenduste** kohta peab olema eraldi aruanne, et oleks võimalik reageerida ning pakkuda neile näiteks paketti, milles on suurem andmete salvestamise maht ja rohkem serveri instantse. Edasise arenduse käigus tuleb see automatiseerida.

4.6.4. Loodud veebirakenduse kuvamiskihi nõuded

Järgnevad nõuded kehtivad moodulile, mida kuvatakse loodud **veebirakenduse lõppkasutajale**.

1. Kui **veebirakendus** on privaatne kuvatakse avalehel sisselogimise vormi.
2. Kui **veebirakendus** ei ole privaatne kuvatakse menüüribal sisselogimise ikooni. Autentimata on võimalik avada ligipääsupiiranguta menüüvalikuid.
3. Kui **veebirakenduses** ei ole määratud ühtegi ligipääsupiirangut siis sisselogimise vormi ega ikooni menüüribal ei kuvata.

4.7. Mittefunktsionaalsed nõuded

1. **Kasutaja** poolt loodud **veebirakendus** peab olema erinevatel ekraanidel skaleeruv ja **kasutaja** poolt paigutatud elementide suhteline asukoht peab ekraaniresolutsiooni muutmisel säilima.
2. Loodava lahenduse **lõppkasutaja** visuaalse kasutajaliidese veebipõhiste komponentide ülesehitus ja disain peavad võimaldama ja toetama nende kasutamist puutetundlikel seadmetel (nutitelefonid, tahvelarvutid jmt).
3. **Platvorm** ja loodud **veebirakendused** peavad olema kättesaadavad 24 tundi ööpäevas.
4. Kuna **platvormi** potentsiaalsed **kasutajad** võivad asuda erinevates ajavööndites tuleb hooldustööde teostamiseks valida ajavahemik, mis mõjutaks **kasutajaid** kõige vähem. Võimaluse korral tuleb hooldustöid teha etappide kaupa selliselt, et katkestus rakenduse töös toimuks ajavahemikus 23:00 – 06:00.
5. Loodavad **veebirakendused** peavad töötama enamlevinud veebibrauserite vähemalt viimase kolme versiooniga.
6. Süsteemi kasutajaliidese funktsionaalsus peab olema **kasutajale** koheselt selge ja arusaadav.
7. **Platvorm** peab olema tõlgitav erinevatesse keeltesse. Võimalusel tuleks kasutada pilte, millel ei ole teksti või kirjutada sellised graafilised osad koodina, et vähendada spetsiifiliselt iga keele jaoks loodavate piltide arvu.
8. Kogu andmevahetus peab toimuma HTTPS-protokolli kasutades.
9. Vead peavad olema logitud.
10. Lubatud reageerimisaeg (*response time*) on 3 sekundit.
11. Kasutajaliides peab olema programmeeritud selliselt, et erinevaid elemente (nupud, vormid, tabelid) rakenduste loomiseks oleks võimalik järk-järgult lisada, vt ptk. 3.5 p.1 „must kast“. Uute elementide lisamine ei tohi mõjutada olemasolevaid elemente. Sama loogika alusel peab olema võimalik eemaldada elemente selliselt, et see ei mõjuta teisi olemasolevaid elemente.
12. Süsteemi siseselt suhtlevad komponendid REST API kaudu.
13. Veebibrauseris kasutamiseks ei ole **kasutajal** vaja teostada allalaadimisi, süsteem on kasutatav veebipõhiselt.
14. Nii operatsioonisüsteemi kui ka andmebaasitarkvara tasemel rakendatakse perioodilist (24 h) automaatset turvakoopiate tegemist vastavate tarkvaratoodete

vahenditega. Turvakoopiate tegemine ei tohi häirida süsteemi tööd tööajal, soovitav on teha täielikke turvakoopiad öösel infosüsteemi tööajast väljapoole jääval ajal.

15. Andmebaasidest tehakse automaatseid perioodilisi (1 kord nädalas) täielikke turvakoopiaid. Lisaks sellele logitakse kõik andmebaaside muudatused vastavate andmebaasitootjate vahenditega, et oleks võimalik taastada iga ajahetke seis. Täiskoopiate tegemine ei tohi häirida süsteemi tööd ja peab olema võimalusel planeeritud (vastava ajavööndi) öisele ajale (23:00 – 06:00).
16. **Platvormi** esimeses versioonis peab olema vähemalt manuaalselt võimalik platvormi haldajal taastada andmebaas soovitud ajahetkele.
17. Iga kasutaja sisse- ja väljalogimine dokumenteeritakse süsteemilogidesse.
18. Andmebaasidesse salvestatavate andmete puhul tuleb lisada kontrollid, et välistada pahatahtlikku rünnet näiteks veebivormidesse sisestatavate andmete kaudu. Sellised elemendid peavad juba vaikimisi kasutajaliideses neid kontrolle sisaldama. Lisaks peavad olema samad kontrollid süsteemi *back-end* poolel.
19. Süsteemil peab olema jälgimissüsteem, et vältida pahatahtlike kasutajate tegevust (näiteks krüptoraha kaevandamine), mis läheb vastuollu kasutatava pilveteenusepakkuja eeskirjade ja tingimustega.
20. **Veebirakendusele** peab saama määrata oma domeeninime.

4.8. Tehnilised piirangud ning nende lahendused

Lähtuvalt loodava platvormi iseärasustest ning kasutajauuringu tulemustest tuleb lahendada mitmeid tehnilisi küsimusi ning leida optimaalseimad lahendused.

4.6.2 Andmebaasitabelid

Loodav platvorm lubab kasutajatel kirjeldada endale sobival kujul tabeleid. Sellise rakenduse puhul ei ole mõistlik piirata kuidagi tabelite loomist kuna see läheks vastuollu platvormi algse ideega lubada kasutajatel vabalt luua endale sobivaid veebirakendusi. Sellega seoses kerkib mitmeid probleeme, millele tuleb lahendus leida. Need probleemid oleksid järgnevad:

1. Relatsioonilises andmebaasis on keeruline lubada kasutajatel ise tabeleid luua või modelleerida andmebaas selliselt, et kasutajatel ei oleks olulisi piiranguid.

2. Kasutajate poolt loodud rakendused võivad olla väga erineva sisu ja eesmärgiga. Osa kasutajaid vajab vaid ühte tabelit, mõni teine aga sadu.
3. Kuidas lahendada erinevate andmemahtude probleem? Kui üks rakendus vajab vaid mõnda megabaiti siis teise rakenduse andmemahtude vajadus võib ulatuda sadadesse gigabaitidesse või isegi rohkem.
4. Kuidas lahendada turvalisuse probleem ehk hoida kõikide rakenduste andmed üksteisest võimalikult turvaliselt lahus?
5. Skaleeruvuse ja asukoha probleem – kliendid võivad asuda erinevates regioonides üle maailma ning omakorda nende kliendid (loodud rakenduste kasutajad) võivad asuda erinevates piirkondades.

Andmebaasisüsteemidest on enamlevinud klassikalised relatsioonilised andmebaasid ning dokument-tüüpi andmebaasid ehk NoSQL. NoSQL andmebaaside erinevus võrreldes relatsiooniliste andmebaasidega on see, et ei ole nõutud kindlat andmebaasiskeemi ehk tabelite struktuuri. Dokumentandmebaasis luuakse objekte ehk dokumente, mis ei pea olema teistega sarnase struktuuriga. NoSQL tüüpi andmebaasi eeliseks on ka selle lihtne skaleeritavus – soovi korral on võimalik samu andmeid hoiustada üle maailma erinevates andmekeskustes, et vähendada lõppkasutaja päringuaegasid. Seetõttu on antud juhul kõige otstarbekam kasutada klientide rakendustes dokument-tüüpi andmebaasi. [22]

Samas on vaja hoiustada ka äripoole andmeid, näiteks platvormi klientide andmed, lepingud, hinnakirjad, arveldused klientidega jne. Selle jaoks on sobiv relatsiooniline andmebaas, näiteks PostgreSQL [23].

Kasutajate rakenduste andmete lahushoidmiseks ja andmemahtude haldamiseks on kaks meetodit. Esimene variant oleks hoiustada kõiki andmeid ühes füüsilises andmebaasis ja nende mahtu pidevalt jälgida ning rakendada piisavaid turvameetmeid, et erinevad rakendused ei pääseks näiteks programmeerija eksituse tõttu üksteise andmetele ligi.

Teine meetod oleks hoiustada iga loodud rakenduse andmeid füüsiliselt erinevates andmebaasides. Sellel variandil on mitu eelist – ei ole vajalik arvutada andmemahtusid, sest ühe andmebaasi maht ongi konkreetse rakenduse poolt kasutatav maht ning üks andmebaas sisaldab vaid ühe rakenduse andmeid. Antud platvormi iseärasusi arvestades

on teine meetod kindlam valik ja lihtsustab ka andmete haldamist. Dokument-tüüpi andmebaasidest on kõige levinum MongoDB [22].

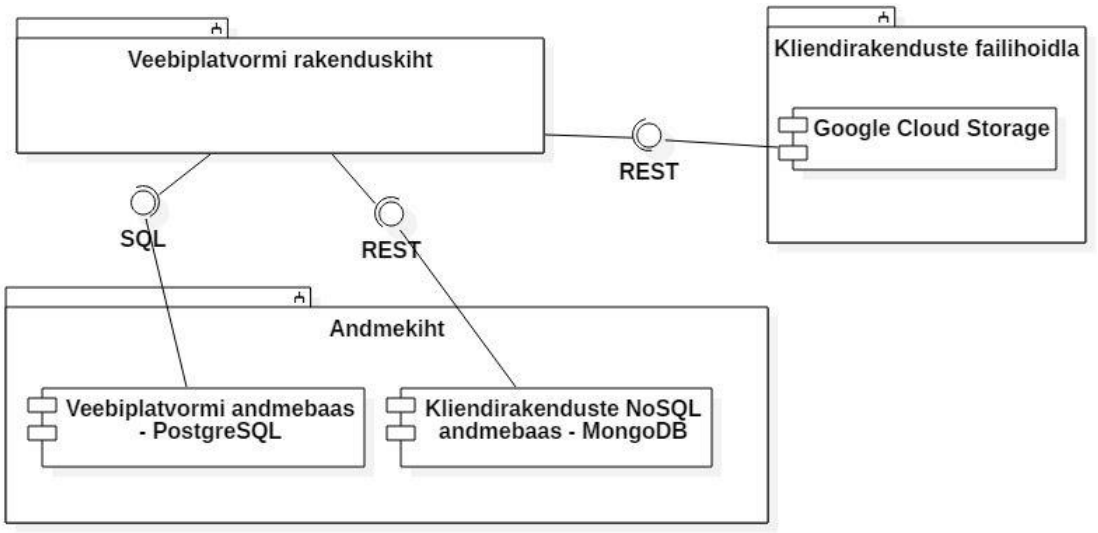
Andmebaaside haldamiseks on omakorda kaks meetodit – kas hallata andmebaase ise enda serverites või kasutada mõnda sellele spetsialiseerunud teenusepakkujat. Tabel 2 toob välja erinevate teenusepakkujate hinnad ühe näidisandmebaasi haldamiseks. Aluseks on võetud mahupiirang 10GB kuna mitmete teenusepakkujate puhul on see minimaalne maht. Andmehulgaks, mida ühes kuus alla- ja üles laetakse on näidispakettide puhul võetud 10GB.

Tabelis märgitud täisteenus on NoSQL andmebaasiteenus, mis sisaldab automaatset haldamist, monitooringut, erinevate regioonide vahel jaotamist ja muid lisafunktsioone. Andmebaasiga on võimalik suhelda näiteks üle API. Virtuaalserver eeldab ise andmebaasi ülesseadmist ning selle haldamist.

Ülevaatest selgub, et kõige kallim on Amazon AWS Document DB teenus, mis on NoSQL andmebaasi täisteenus. Samas on näiteks Google Cloud Datastore teenuse hind võrreldav Amazon AWS EC2 ja Digital Ocean Droplet virtuaalserverite pakettidega. Kuna arendatava platvormi peamine sihtrühm on väikeettevõtted, mille eelarved ei ole piisavalt suured, et tellida mahukat tarkvaraarendusteenust siis võib eeldada, et need ettevõtted on ka hinnatundlikud igakuiste kulude osas. Seetõttu tuleb lahenduste valimisel lähtuda kindlasti ka teenuse hinnast. Kuigi virtuaalserverite renditasu on madalam on nende haldamine töömahukam. Seetõttu on antud juhul kõige mõistlikum valik Google Cloud Datastore, mille puhul ei ole serveri administreerimist vaja. Lisaks lahendab selline teenus skaleeruvuse ja regioonide probleemi – Google Cloud pilvepõhise teenuse kasutamisel on lihtne valida iga rakenduse jaoks geograafiliselt kõige lähemal asuvaid andmekeskuseid ning vajadusel enda rakenduse andmeid andmekeskuste vahel dubleerida. Lisaks sellele on võimalik valida andmekeskuse asukohta vastavalt konkreetse kasutaja rakendusele, et selle rakenduse andmete hoiustamine vastaks GDPR määrusele [24].

Kuna NoSQL andmebaasid ei võimalda salvestada suuri faile (MongoDB dokumendi maksimaalne lubatud maht on 16 megabaiti [25]) siis tuleb paralleelselt kasutada ka failide hoiustamiseks mõeldud teenust. Google pilveteenuste lahendustes on selleks olemas pakett Google Cloud Storage, mida kasutab näiteks Spotify muusikastriimimise

teenus enda audiofailide hoiustamiseks [26]. Selle teenuse maksumus on kuni 0,036 dollarit/GB kohta ehk 10 GB mahu korral oleks kuutasu koos andmete edastamisega umbes 1,53 dollarit (1,36 eurot) [27]. Olenevalt rakenduse eripärast võib andmete hoiustamise tasu muutuda kasutajale eelnevast arvestusest soodsamaks – kui hoiustatakse palju suuremahulisi faile aga vähe andmebaasis hoiustatavaid andmeid siis muutub kallim, NoSQL andmebaasiteenuse kuutasu soodsamaks ning kasutatakse pigem odavamat, failide hoiustamise teenust. Joonis 14 kujutab komponentdiagrammil andmebaaside ja failihoidla osa muu süsteemi suhtes.



Joonis 14 Andmebaaside komponentdiagramm (Allikas: autori koostatud).

Tabel 2 NoSQL andmebaasi teenusepakkujate ülevaade (Allikas: autori koostatud).

Teenuse-pakkuja	**Amazon AWS**	**Microsoft Azure**	**Google Cloud**	**Amazon AWS**	**MongoDB, Inc.**	**Digital Ocean**
Teenuse nimetus	Amazon DocumentDB	Cosmos DB	Cloud Datastore	Amazon EC2	MongoDB Atlas	Droplets
Tüüp	Täisteenus	Täisteenus	Täisteenus	Virtuaalserver	Täisteenus	Virtuaalserver
Märkused	200 miljonit päringut kuus	400 päringut sekundis	200 miljonit päringut kuus			1GB RAM, 1 CPU, 25GB SSD
Veebileht	https://aws.amazon.com/documentdb/	https://azure.microsoft.com/en-us/services/cosmos-db/	https://cloud.google.com/datastore/	https://aws.amazon.com/ec2/?nc2=h_m1	https://www.mongodb.com/cloud/atlas	https://www.digitalocean.com/products/droplets/
Kuutasu	$238,52/€212,11	$26,50/€23,57	$10,42/€9,27	$16,45/€14,63	$58,00/€51,58	$5,00/€4,45

4.6.2 Loodud rakenduste haldamine ja hoiustamine

Sarnaselt andmebaasidele on ka kasutajate koostatud rakenduste osas mitmeid lahendamist vajavaid probleeme.

1. Kasutajate poolt loodud rakendused võivad olla väga erineva sisu ja eesmärgiga. Osa kasutajaid vajab väikest arvutusvõimsust, teine osa aga suuremat.
2. Erinevate andmeedastusmahtude probleem - üks rakendus võib vajada vaid murdosa mõne teise rakenduse tööks vajalikust andmeedastuse mahust. Näiteks videofailide edastusteenus võrreldes kliendiandmete hoiustamisega.
3. Kuidas lahendada turvalisuse probleem ehk hoida kõikide rakenduste andmed üksteisest võimalikult turvaliselt lahus?
4. Kuidas viia loodud rakendus selle kasutajatele füüsiliselt võimalikult lähedale?

Analüüsitava platvormi üks iseärasusi on see, et hallatakse palju veebirakendusi, mille eesmärk ja kasutajate geograafiline asukoht on erinevad. Iga rakenduse jaoks individuaalse andmebaasi instantsi loomisega (vt. ptk. 4.6.1.) on lahendatud probleem, et andmeid saaks hoiustada lõppkasutajale võimalikult lähedal asuvas andmekeskuses. Kuna kasutajad loovad ise veebirakendusi tähendab see seda, et ka nende veebirakenduste koodibaas tuleb kuskile salvestada ehk ei ole võimalik kasutada traditsioonilist ühte koodibaasi kõikide kasutajate jaoks. Kuna veebirakenduste jaoks luuakse igal juhul andmebaasi instants, mille asukohta on mugav määrata, on mõistlik ka kogu loodud veebirakendusepõhine kood salvestada samasse andmebaasi. Programmikood, mis oskab lugeda kasutaja andmebaasist tema loodud rakenduse andmed ja sellest genereerida kuvamiskihi ehk lõpliku veebirakenduse, saab olla kõigi jaoks ühine.

Loodavaid veebirakendusi tuleb hoida üksteisest piisavalt lahus, et tagada turvalisus ning jagada neile erinevat arvutus- ning võrguliikluse ressurssi. Ressursside jagamise ning monitoorimise lihtsustamiseks tuleb sarnaselt andmebaaside lahendusele hoida ka loodavad veebirakendused füüsiliselt eraldatuna.

Kuna antud juhul tuleb arvestada rahalise kuluga ning sellega, et tõenäoliselt ei nõua loodavad veebirakendused suuri arvutusvõimsusi siis on nende tingimuste tagamiseks kaks meetodit – virtuaalsed pilveserverid või vastavad serverita (*serverless*) teenused. Suuremate pilveteenusepakkujate puhul on reeglina tavalised virtuaalserverid tunduvalt odavamad, kui spetsiaalsed teenused. Lisaks on nende haldamine läbi vastavate API-de

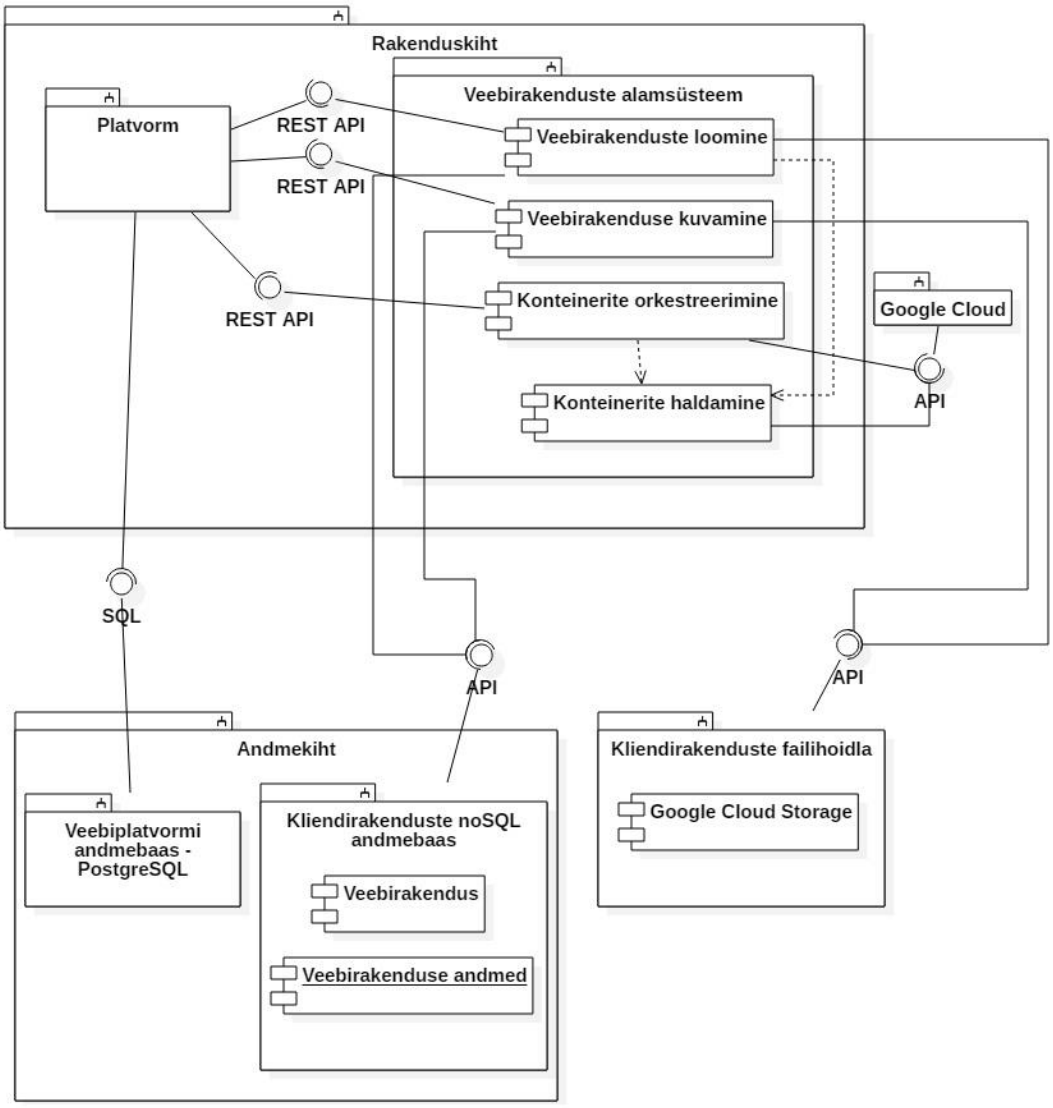
tehtud piisavalt mugavaks, et virtuaalserverite loomist on võimalik automatiseerida. Näiteks Google Cloud Compute Engine teenuses, mis võimaldab luua virtuaalservereid, on kõige odavama serveri hind 4,68 dollarit (4,16 eurot) kuus. See server sisaldab jagatud protsessorit, 0,6GB muutmälu ning 10GB salvestusmahtu. Digital Oceani Droplet teenuse puhul on ühe virtuaalserveri hind 5 dollarit (4,45 eurot) kuus (vt. Tabel 2). Tõenäoliselt suudab selline server ära teenindada mitu madala aktiivsusega veebirakendust.

Virtuaalserverite haldamiseks tuleb luua platvormile eraldi komponent, mis sellega tegeleb. Konkreetse teenuse valik sõltub tulevastest ärianalüüsidest ning tõenäoliselt teenuste hindadest. Süsteemi arhitektuuris on igal juhul vajalik eraldi komponent – tavaliste virtuaalmasinate puhul on sellise komponendi loomine keerukam kuna nõuab rohkem seadistamist, täisteenuste puhul on selle komponendi loomine aga tõenäoliselt ka lihtsam. Seetõttu tuleb nende otsuste langetamisel arvestada kindlasti ka töömahtu ning haldamisele kuluvat ressurssi.

Et loodud veebirakendusi lahus hoida ning neid mugavalt hallata on mõistlik kasutada konteineriseerimise tehnoloogiat. Docker [28] on kõige levinum konteinerite loomise vahend. Konteiner on standardiseeritud kokku pakitud tarkvara, mida on võimalik mugavalt liigutada erinevate keskkondade vahel ilma, et tekiks vajadus uut keskkonda seadistada või muul viisil selle tarkvaraga ühildada. Suure hulga konteinerite orkestreerimiseks on kõige levinum tarkvara Kubernetes [29], mis võimaldab lihtsasti konteinereid dubleerida ning jagada erinevate serverite vahel. Lisaks võimaldab see teha korraga paljudes konteinerites vajalikke tegevusi ning automaatselt konteinereid vastavalt vajadusele luua või kustutada.

Seega saab vajadusel klientide veebirakendusi lihtsasti dubleerida, luues uusi konteinereid ja soovi korral hoiustada neid erinevates regioonides. Konteinerite metoodika annab võimaluse ka lihtsasti teha hinnaarvestusi (platvormi kasutamise eest) ning jälgida väga täpselt loodud veebirakenduse jõudlust.

Joonis 15 kajastab rakenduskihi ja andmekihi komponente. Antud joonisel on „Platvorm“ alamsüsteem, mille ülesandeks on tegeleda klientide andmetega (arveldused, lepingud) ja suunata päringud õige veebirakenduseni. Veebirakenduste alamsüsteemi ülesanne on veebirakenduste loomine, nende kuvamine lõppkasutajale ning konteinerite haldamine.



Joonis 15. Rakendus- ja andmekihi komponentdiagramm (Allikas: autori koostatud)

5. Lahenduse kirjeldus

Eelneva analüüsi põhjal on võimalik luua lahenduse kirjeldus. Äripoolt kirjeldavad ärireeglid, üldine kirjeldus ning kasutusmallid. Arhitektuurilise poole osas kirjeldatakse süsteemi komponentide omavahelisi seoseid ja loogikat, andmemudeleid ning paigaldust evitusdiagrammina. Tegemist on platvormi loomise esimese etapiga ning seetõttu ei ole lahenduse kirjeldamisel mindud väga detailidesse. Täpsemad analüüsid, eriti äri poolelt, tehakse projekti järgmises etapis.

5.1 Ärikirjeldus

Loodav veebiplatvorm on mõeldud eelkõige arvutikasutajatele, kes ei ole programmeerijad, kuid sooviksid ise luua endale sobivaid veebirakendusi. Loodavate veebirakenduste kasutusvaldkond ei ole piiratud ning platvorm peab võimaldama luua kasutaja soovidele vastavat lahendust.

Süsteemi kasutamiseks ei ole vaja eraldi väljaõpet ning selle kasutajaliides on piisavalt intuitiivne, et keskmine arvutikasutaja oskaks lihtsamaid rakendusi ise luua.

Veebirakenduste loomiseks tuleb kasutajal registreerida endale kasutajakonto. Soovi korral on võimalik lisada kohe ka makseandmed, kuid tasuta paketi kasutamiseks ei ole see vajalik. Tasulise paketi kasutamise korral toimub kasutaja krediitkaardikontolt arveldamine igakuiselt automaatselt. Paketi kasutamise lõppemisel ei ole võimalik loodud veebirakendusi enam kasutada ning neid teenindavad virtuaalserverid suletakse. Rakenduse andmed säilitatakse pärast kasutusaja lõppu piiratud aja jooksul ning kasutajal võimaldatakse nendest andmetest ka digitaalsel kujul varukoopiaid teha.

Uue veebirakenduse loomiseks ei ole programmeerimise oskused vajalikud ning platvormi kasutajaliides võimaldab visuaalse programmeerimise abil kirjeldada ning luua sobiv veebirakendus. Kasutaja saab luua valmiskomponentide abil sobiva kasutajaliidese enda veebirakendusele. Andmete haldamiseks on võimalik kirjeldada andmebaasitabelid. Kasutusmugavuse ja -lihtsuse säilitamiseks on tabelite kasutamine sarnane enamlevinud kontoritarkvarade tabeltöötlusprogrammidega. Kasutajaliidese loomisel saab kasutaja valida, milliseid andmeid konkreetsest tabelist kuvatakse, muudetakse või kustutatakse.

Lisaks veebirakenduste loomisele saab kasutaja jälgida temale reserveeritud virtuaalserverite jõudlust, andmebaasi täituvust ning võrgukoormust. Vajadusel on tal võimalik administreerimise moodulis tellida juurde serveri instantse, suurendada andmebaasi jaoks vajaminevat mahtu ning muuta nende instantside asukohta regioonide lõikes.

Lõppkasutaja saab veebibrauseri abil külastada loodud veebirakendust. Kui veebirakenduses on määratud ligipääsuõigused peab ta end autentima e-maili aadressiga. Esmakordsel registreerimisel saadetakse e-mailile piiratud kehtivusajaga link, et luua parool. Enamlevinud autentimisviise kasutades (Facebook, Google) ei pea parooli määrama.

Platvormi haldajatel on administraatori vaade, kuid andmetele ligipääs on erinevate õigustega reguleeritud. Kõige suuremate õigustega on administraatori õigustes kasutaja. Administraatori õiguseid saab anda vaid teine administraator.

Administraatori vaates on võimalik tutvuda klientide andmetega (kontaktid, lepingud) ning näha samuti kliendi rakenduste tehnilist infot - virtuaalserverite jõudlust ja arvu, andmebaasi täituvust ning võrgukoormust. Administraatori vaade võimaldab teha samu tegevusi, mida kasutaja rakenduste haldamise vaade ehk lisada juurde serverite instantse, et parandada jõudlust või jaotada servereid erinevate regioonide ja suurendada andmebaasimahtu. Selle eesmärk on võimaldada pakkuda kasutajatele kasutajatuge selliselt, et platvormi haldaja saaks osa tegevusi ka kasutaja eest teha.

Vastavate õiguste olemasolul saab platvormi haldaja esindaja muuta ka kasutajate rakendusi. Selle eesmärgiks on eelkõige kasutajatoe pakkumine.

Administraatori vaates on võimalik hallata ka lepinguid, makseinfot ja saata kasutajatele sõnumeid (e-postile).

5.2 Ärireeglid

Ärireeglid seavad tingimused ja piirangud erinevatele äriprotsessidele. Sellega seatakse piirangud ka süsteemile ning need võimaldavad täpsemalt luua ning hinnata kasutuslugusid. Analüüsitava platvormi ärireegleid kajastab Tabel 3.

Tabel 3. Ärireeglid (Allikas: autori koostatud)

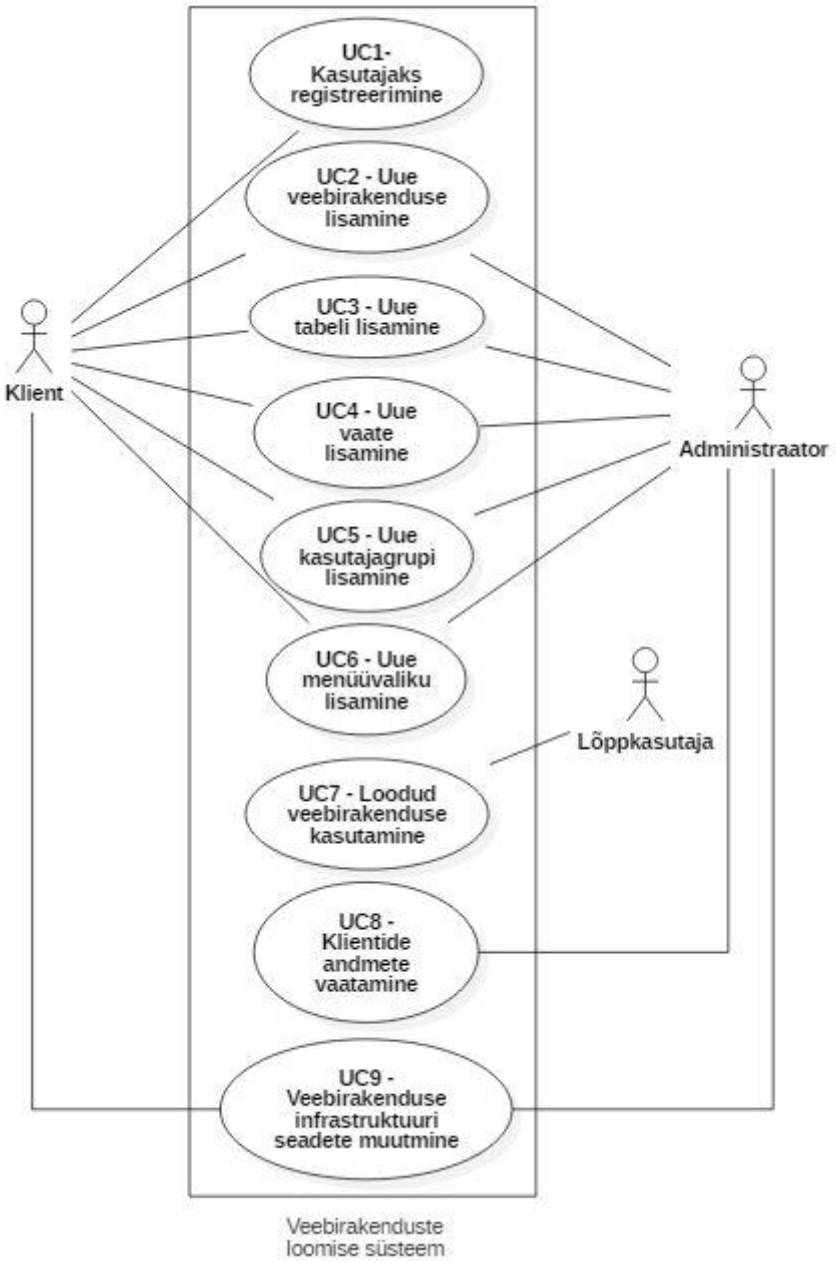
ID	Ärireegel
R1	Süsteemi kasutajaid on kolme tüüpi – administraatorid, veebirakenduste loojad (kliendid) ning veebirakenduste kasutajad (lõppkasutajad).
R2	Igal kasutajal võib olla üks kasutajakonto.
R3	Üks kasutaja võib olla seotud erinevate veebirakendustega.
R4	Veebirakenduste haldamisel peab saama igale kasutajale määrata erinevaid õiguseid.
R5	Administraatori rollis kasutaja saab vaadata kõikide loodud veebirakenduste tehnilisi andmeid.
R6	Administraator tüüpi kasutajatele peab saama määrata erinevaid õiguseid.
R7	Administraator tüüpi kasutaja saab vastavate õiguste olemasolul muuta klientide veebirakenduste andmeid.
R8	Administraator tüüpi kasutaja näeb kõikide klientidega seotud andmeid koondvaates.
R9	Administraator tüüpi kasutaja saab süsteemis lisada, muuta või kustutada lepinguid.
R10	Klient saab süsteemis lisada, muuta või kustutada endaga seotud lepingut.
R11	Administraator tüüpi kasutaja saab filtreerida süsteemis klientide andmebaasi ning teha saadud andmetest väljavõtteid.
R12	Administraator tüüpi kasutaja saab saata süsteemist masspostitusena klientidele HTML-vormingus e-maile.
R13	Veebirakenduse looja saab lisada kasutajaid, kes saavad samuti rakendust hallata.
R14	Kui loodud veebirakendus on ligipääsupiiranguga siis saavad seda vaadata vaid vastava ligipääsuga lõppkasutajad.

5.3 Kasutusmallid

Kasutusmallid kirjeldavad täpsemalt süsteemi funktsionaalsust, mida väline aktor süsteemilt ootab. Igal kasutusmallil peab olema konkreetne tulemus. Analüüsitaval platvormil on kolm aktorit – administraator, klient ja lõppkasutaja. Käesolevas töös on välja toodud peamised kasutusmallid, projekti järgmistes etappides tuleb teha detailsem analüüs leidmaks täpsemaid nõudeid ja täiendavaid kasutusmalle. Kasutusmallid on välja toodud lisas 3.

5.4 Kasutusmallide mudel

Kasutusmallide mudel kujutab erinevate aktorite seoseid kasutusmallidega. Süsteemis on kolm aktorit – klient, administraator ning lõppkasutaja. Klient koostab veebirakenduse, administraatoril on ligipääs samadele funktsioonidele ning lisaks veel platvormi administratiivsetele moodulitele. Lõppkasutaja kasutab loodud veebirakendust. Lõppkasutaja rollis on ka klient, kui ta kasutab rakenduse vaatamise funktsionaalsust.

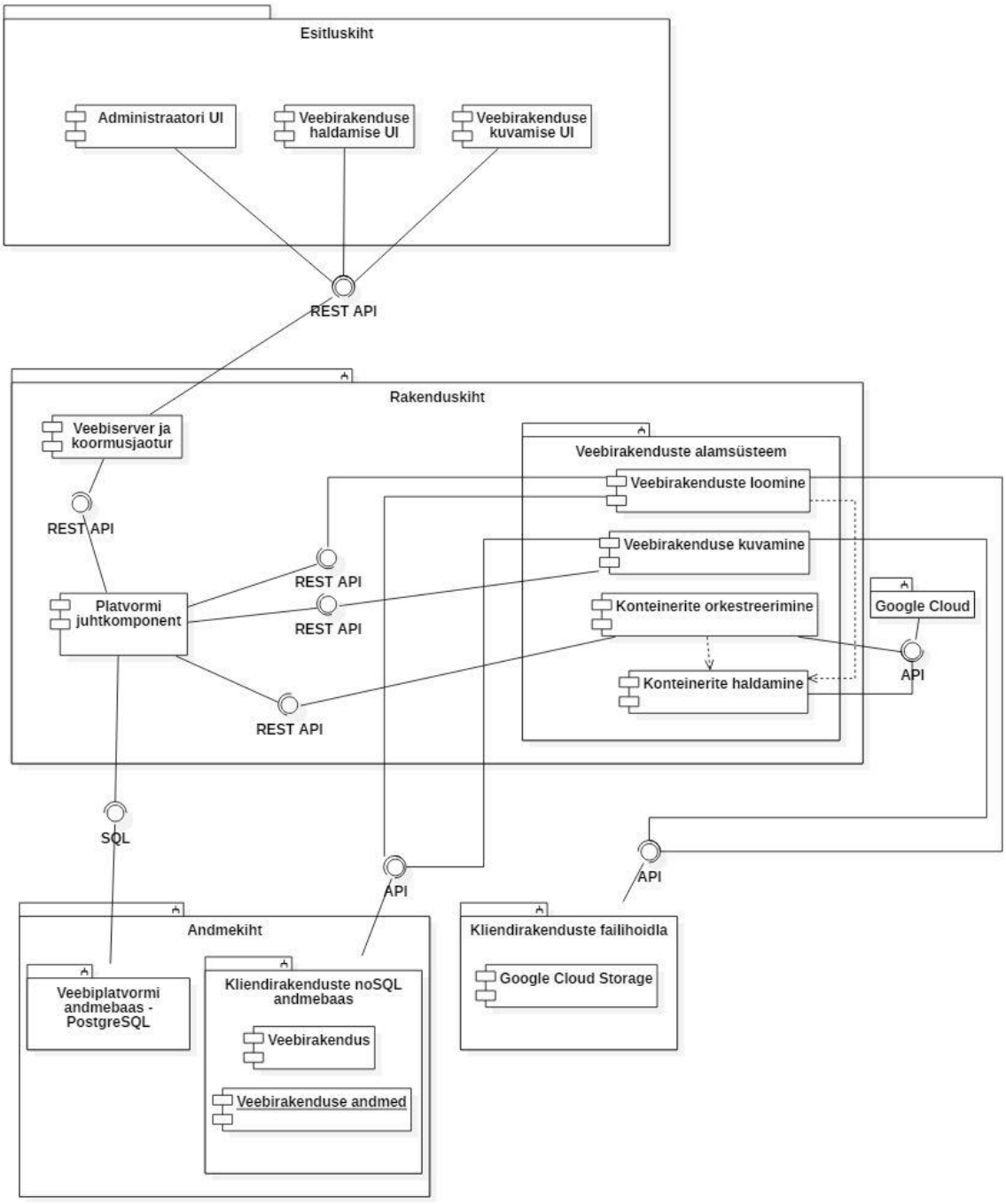


Joonis 16. Kasutusmallide mudel (Allikas: autori koostatud).

5.5 Komponentdiagramm

Arhitektuuriliste lahenduste valikul võeti arvesse tehnilised iseärasused ja nendest tulenevad piirangud (vt. ptk. 4.6.2. ja 4.6.3.). Lisaks tuleb arvestada funktsionaalsete- ja mittefunktsionaalsete nõuetega.

- **Administraatori UI –** Kasutajaliidese komponent administraatori mooduli kuvamiseks.
- **Veebirakenduse haldamise UI** – Kasutajaliidese komponent veebirakenduste loomiseks ja haldamiseks.
- **Veebirakenduse kuvamise UI** – Eraldi komponent valminud veebirakenduste kuvamiseks lõppkasutajale.
- **Platvormi juhtkomponent** – Platvormi põhiloogikat sisaldav komponent. Vastavalt saabunud päringule võtab ühendust veebirakenduste loomise, kuvamise või orkestreerimise komponentidega. Lisaks edastab eelnevalt nimetatud komponentidele andmeid rakenduse ja kasutajate kohta.
- **Veebirakenduste loomine –** Veebirakenduste loomise ja haldamise back-end komponent.
- **Veebirakenduste kuvamine –** Veebirakenduste kuvamise back-end komponent.
- **Kontenerite orkestreerimine** –Kubernetese komponent konteinerite haldamiseks.
- **Konteinerite haldamine** – Komponent veebirakenduste jaoks konteinerite loomiseks ja kustutamiseks.
- **Google Cloud –** Google Cloud pilveteenus konteinerite hoiustamiseks.
- **Veebiplatvormi andmebaas (PostgeSQL)** – Andmebaas platvormi andmete hoiustamiseks (klientide andmed, rakenduste metaandmed, arveldused).
- **Kliendirakenduste noSQL andmebaas** – Dokument-tüüpi andmebaas veebirakenduste ja nende andmete hoiustamiseks.
- **Google Cloud Storage** – Pilveteenus suuremahuliste failide hoiustamiseks.



Joonis 17. Platvormi komponentdiagramm (Allikas: autori koostatud).

5.6 Andmemudel

Süsteemis on kasutusel kahte tüüpi andmebaase. Relatsiooniline andmebaas on platvormi üldiste andmete hoiustamiseks (kliendid, lepingud, arveldused). Igal loodud veebirakendusel on noSQL andmebaas selle konkreetse rakenduse koodi ning andmete hoiustamiseks. Joonis 30 kujutab platvormi relatsioonilise andmebaasi füüsilist andmemudelit. Tabel 14 kirjeldab andmemudeli tabelite semantikat.

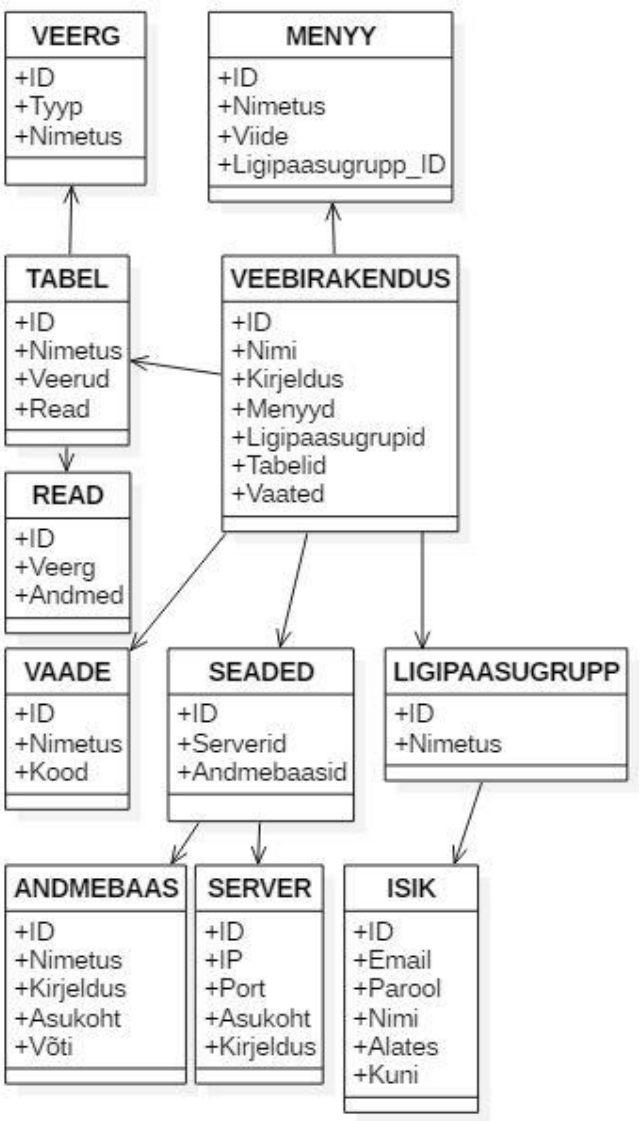
NoSQL- ehk dokumentandmebaas erineb oma olemuselt relatsioonilsest andmebaasist selle poolest, et puuduvad andmebaasitabelid ja kindlad seosed erinevate tabelite vahel. Andmeid salvestatakse dokumentidena ning erinevate dokumentide struktuur ei pea vastama samale skeemile. Kuna noSQL ehk dokumentandmebaasi ei saa kirjeldada relatsioonilise andmemudeliga on üheks lahenduseks kirjeldada andmeid ülem- ja alamobjektidena ning nendevaheliste seostega.

Joonis 18 kirjeldab kasutajate poolt loodavate veebirakenduste andmebaasi loogikat ning sisu. Dokumentandmebaasis salvestatakse ülemobjekt „Veebirakendus“, mis sisaldab selle rakendusega seotud alamobjekte. Tabel 4 kajastab nende objektide semantikat.

Tabel 4. noSQL andmebaasi andmemudeli andmeobjektide semantika (Allikas: autori koostatud).

Andmeobjekt	**Semantika**
VEEBIRAKENDUS	Põhikomponent, mis sisaldab kõiki kasutaja poolt loodud rakenduse andmeid. Lisaks rakenduse nimele ja kirjeldusele sisaldab alamkomponente menüüde, ligipääsugruppide, tabelite ja vaadete jaoks.
MENYY	Andmeobjekt menüüde salvestamiseks. Sisaldab menüü nime, viidet vaatele (lehele) ja ligipääsugrupile, kui menüüvalikule ligipääs on piiratud.
TABEL	Veebirakenduse tabelite nimekirja üks objekt. Sisaldab tabeli nimetust ja ridade ning veergude infot alamobjektidena.
VEERG	Sisaldab tabeli veergude ehk atribuutide kirjeldusi – pealkiri ja andmetüüp, mida selles veerus kasutada tohib.
READ	Tabeli ridade andmed. Sisaldab viidet veerule ning andmeid, mida kasutaja või lõppkasutaja rakendusse salvestavad.
VAADE	Loodud veebirakenduse vaadete (lehtede) andmed. Nimetus ja JSON-kujul elementide nimekiri ning nende asukoha kirjeldus.
SEADED	Veebirakenduse seadete objekt. Sisaldab alamobjekte rakenduse andmebaasi ja serveri (konteineri) kohta.
ANDMEBAAS	Veebirakendusega ühendatud andmebaasi info. Sisaldab nimetust, asukohta ja andmebaasiga ühendamise andmeid.
SERVER	Veebirakenduse serveri (konteineri) info. Sisaldab näiteks IP-aadressi, pordi ja asukoha infot.
LIGIPAASUGRUPP	Kasutaja poolt veebirakendusse lisatud kasutajagruppide info. Sisaldab kasutajagrupi nimetust ning alamobjektina sellesse gruppi kuuluvaid isikuid.

Andmeobjekt	Semantika
ISIK	Ühte ligipääsugruppi kuuluvate isikute haldamiseks mõeldud andmeobjekt. Sisaldab isiku nime ning ligipääsuks vajalikke e-posti aadressi ja parooli.



Joonis 18. Kliendirakenduste noSQL andmebaasi mudel (Allikas: autori koostatud).

5.7 Evitusdiagramm

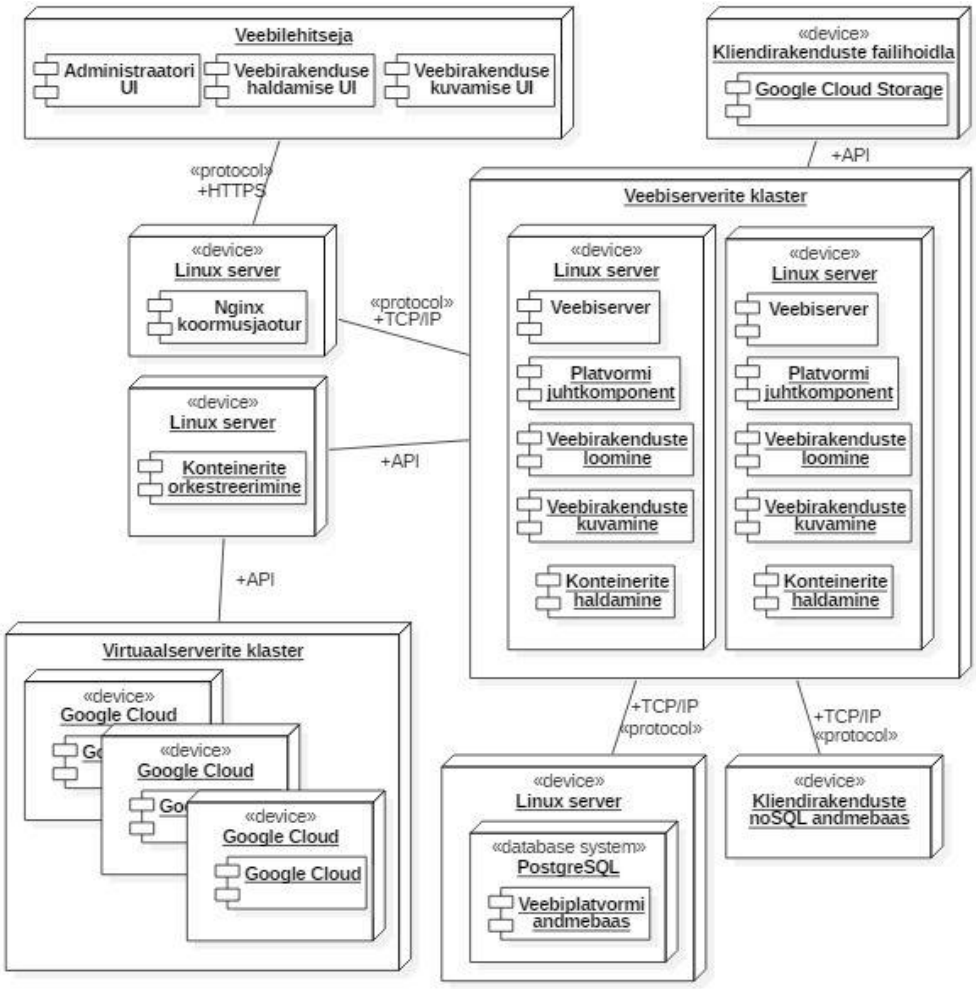
Evitusdiagramm (Joonis 19) kirjeldab süsteemi erinevate osade füüsilist paigutamist riistvarale. Analüüsitava süsteemi komponendid tuleb erinevatel põhjustel paigutada mitmetele erinevatele riistvarakomponentidele.

Veebilehitseja asub lõppkasutaja arvutis ning kasutab vastavat komponenti veebirakenduse kasutajaliidese kuvamiseks. Eraldi server on reserveeritud koormusjaoturi, konteinerite orkestreerimise (Kubernetes) ning veebiplatvormi relatsioonilise andmebaasi jaoks. Koormusjaotur asub eraldi serveris kuna seda läbib kogu võrguliiklus ning vajadusel on võimalik konkreetse serveri ressursse suurendada. Konteinerite orkestreerimise komponent töötab iseseisvalt ning seda ei ole eeldatavalt vaja dubleerida. Samas vajab ta piisavat ressurssi tõrgeteta toimimiseks. Relatsiooniline andmebaas asub samuti eraldi serveris kuna seda ei jagata ega dubleerita erinevate tsoonide vahel.

Veebiserverite klaster koosneb dubleeritud serveritest, mis sisaldavad platvormi juhtkomponenti ning veebirakenduste loomisega seotud komponente. Neid servereid on võimalik lihtsasti kloonida ning lisada juurde vastavalt ressursinõudlusele või lõppkasutaja asukohale, et olla kasutajale füüsiliselt võimalikult lähedal.

Virtuaalserverite klaster sisaldab endas virtuaalmasinaid, mis omakorda sisaldavad loodud veebirakenduste konteinereid. Neid virtuaalservereid peab olema võimalik lisada vastavalt kasutajate seadistustele – kui kasutaja soovib määrata enda veebirakendusele rohkem ressurssi või hoiustada rakendust erinevates tsoonides peavad vastavad virtuaalserverid olemas olema.

Kliendirakenduste failihoidla ja noSQL andmebaas on välised teenused ning seetõttu kujutatud samuti eraldi riistvaralisel platvormil.



Joonis 19. Analüüsitava süsteemi evitusdiagramm (Allikas: autori koostatud).

6. Kokkuvõte

Käesolevas magistritöös on välja toodud analüüs ja arhitektuuriline vaade veebiplatvormile, mis võimaldab selle kasutajatel luua visuaalse programmeerimise vahendeid kasutades endale sobilikke spetsiifilisi pilvepõhiseid rakendusi.

Magistritöö eesmärgiks oli viia läbi selle süsteemi analüüs ning pakkuda välja lahendused, et töö tulemusena valminud projekti põhjal oleks võimalik alustada nimetatud platvormi arendamist.

Töö käigus kaardistati kavandatava infosüsteemi nõuded ja tehti ettepanekud tehniliste lahenduste osas. Nõuete kaardistamiseks viidi läbi kasutajauuring küsitluse vormis. Kasutajauuringu ning edasise analüüsi käigus leiti vastused ülesandepüstituses tõstatatud küsimustele.

Kasutajauuringu tulemusel kasutaks 42% küsitletutest sellist platvormi endale sobiliku tarkvara loomiseks. Keskmisest rohkem kasutaksid sellist platvormi keskastme- ja tippjuhid, kes puutuvad rohkem kokku protsesside automatiseerimise vajadusega. Uuringu tulemus tõi välja, et 63% vastanutest on valmis panustama 5 tundi kuni 1 kuu uue tarkvara koostamisele. Seega on kasutajad valmis piisavalt panustama enda aega, et õppida platvormi kasutama ning luua endale vajalikke rakendusi.

Erinevad uuringud on välja toonud visuaalse programmeerimise eelised lõppkasutaja programmeerimise valdkonnas (vt. ptk. 3.4). Lisaks paremale tajule ja kasutusmugavusele on visuaalse programmeerimise eeliseks tihti ka valmiskomponentide kasutamisest tulenev ajaline võit. Seega on mõistlik kasutada sellise platvormi puhul just visuaalset programmeerimist.

Turvalisuse seisukohast rakendatakse platvormil lisaks tavapärastele turvalisuse põhimõtetele ka loodud rakenduste ja andmebaaside füüsilist eristamist. Loodavate veebirakenduste andmebaasid asuvad pilveteenuses ning iga rakenduse jaoks luuakse uus instants. Veebirakenduste tarkvaraline pool asub Dockeri konteinerites ning need on jaotatud erinevate serverite vahel.

Rakenduste hoiustamine Dockeri konteinerites ning veebirakenduste andmebaaside puhul pilveteenuse kasutamine annavad võimaluse ressursside kasutamise lihtsamaks monitooringuks ning vajadusel ressursside lisamiseks läbi uute konteinerite paigaldamise. See aitab tagada selle, et erinevatel rakendustel oleks alati piisavalt serveri ressursse.

Andmekaitsenõuetele vastavuse tagamiseks saab veebirakenduse looja valida enda andmebaasile füüsilise asukoha. Andmete hoiustamise turvalisusega seotud küsimused on lahendatud andmebaasiteenuse pakkuja poolt.

Töö teises osas läbi viidud analüüsi tulemusena kirjeldati loodava veebiplatvormi lahendust.

Ärianalüüsi tulemid olid järgnevad:

- Ärikirjeldus
- Ärireeglid
- Kasutusmallid ja kasutusmallide mudel

Süsteemianalüüsi tulemid olid järgnevad:

- Komponentdiagramm
- Andmemudelid
- Evitusdiagramm
- Veebirakenduse loomise kasutajaliidese prototüüp

Magistritöö autori hinnangul said püstitatud eesmärgid edukalt täidetud ning loodud dokumentatsiooni alusel on võimalik alustada veebiplatvormi esmase versiooni arendamisega.

Projekti järgmistes etappides tuleb esmalt valmis programmeerida platvormi minimaalne elujõuline toode, et oleks võimalik jätkata täpsemate kasutajauuringutega. Lisaks tuleb teha antud töö skoobist välja jäänud detailsem turvaanalüüs.

Kasutatud kirjandus

[1] A. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. Rosson, G. Rothermel, M. Shaw ja S. Wiedenbeck, „The state of the art in end-user software engineering," *ACM Computing Surveys (CSUR),* nr 43, pp. 1-44, 01 aprill 2011.

[2] D. Lizcano, F. Alonso, J. Soriano ja G. López, „Web-centred end-user component modelling," *Future Generation Computer Systems,* nr 54, pp. 16-40, jaanuar 2016.

[3] J. A. Ginige, B. D. Silva ja A. Ginige, „Towards End User Development of Web Applications for SMEs: A Component Based Approach," *International Conference on Web Engineering,* nr 3579, pp. 489-499, 2005.

[4] R. R. Panko, „Spreadsheet Errors: What We Know. What We Think We Can Do.," *Proc. European Spreadsheet Risks Int. Grp,* pp. 7-17, 2000.

[5] Y. Valsamakis ja A. Savidis, „Visual end-user programming of personalized AAL in the internet of things," *Ambient Intelligence,* nr 10217, pp. 159-174, 2017.

[6] M. Noone ja A. Mooney, „Visual and textual programming languages: a systematic review of the literature," *Journal of Computers in Education,* nr 5, p. 149–174, juuni 2018.

[7] „VPL Introduction|Microsoft Docs," Microsoft Corporation, 2012. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/previous-versions/microsoft-robotics/bb483088(v=msdn.10). [Kasutatud 18 veebruar 2019].

[8] J. H. Greyling, C. B. Cilliers ja A. P. Calitz, „B#: The Development and Assessment of an Iconic Programming Tool for Novice Programmers," *2006 7th International Conference on Information Technology Based Higher Education and Training,* pp. 367 - 375 , 2006.

[9] P. P. Ray, „A Survey on Visual Programming Languages in Internet of Things,“ *Scientific Programming,* märts 2017.

[10] O. Díaz, C. Arellano, I. Aldalur, H. Medina ja S. Firmenich, „End-User Browser-Side Modification of Web Pages,“ *Lecture Notes in Computer Science,* nr 8786, pp. 293-307, 2014.

[11] M. Mernik, J. Heering ja A. M. Sloane, „When and how to develop domain-specific languages,“ *ACM Computing Surveys (CSUR),* nr 4, pp. 316-344, Detsember 2005.

[12] T. B. Hilburn, „A top-down approach to teaching an introductory computer science course,“ *SIGCSE '93 Proceedings of the twenty-fourth SIGCSE technical symposium on Computer science education,* nr 25, pp. 58-62, 1993.

[13] D. Scanlan, „Structured flowcharts outperform pseudocode: an experimental comparison,“ *IEEE Software,* kd. 6, nr 5, pp. 28-36, september 1989.

[14] G. Xue, Q. Yang ja J. Xing, „A survey of graphical programming language and its applications in intelligent buildings,“ *2017 Chinese Automation Congress (CAC),* jaanuar 2018.

[15] T. Shimomura, „Visual design and programming for Web applications,“ *Journal of Visual Languages and Computing,* nr 16, pp. 213-230, 2005.

[16] „tadabase.io,“ Tadabase, LLC, [Võrgumaterjal]. Available: https://tadabase.io/. [Kasutatud aprill 2019].

[17] „bubble.is,“ Bubble Group, Inc., [Võrgumaterjal]. Available: https://bubble.is. [Kasutatud aprill 2019].

[18] „Bootstrap,“ [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud aprill 2019].

[19] J. Arnowitz, M. Arent ja N. Berger, Effective Prototyping for Software Makers, San Francisco: Elsevier, Inc., 2007.

[20] „Figma,“ Figma, Inc., [Võrgumaterjal]. Available: https://www.figma.com/. [Kasutatud aprill 2019].

[21] M. Kuniavsky, Observing the User Experience : A Practitioner's Guide to User Research, San Francisco: Elsevier Science & Technology, 2003, p. 560.

[22] „What is NoSQL? | Nonrelational Databases, Flexible Schema Data Models | AWS,“ Amazon Web Services, Inc., [Võrgumaterjal]. Available: https://aws.amazon.com/nosql/. [Kasutatud 25 aprill 2019].

[23] „PostgreSQL: The world's most advanced open source database,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/. [Kasutatud 14 mai 2019].

[24] Riigikogu, „Isikuandmete kaitse seadus,“ 13 mai 2019. [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/104012019011.

[25] „Documents - MongoDB Manual,“ MongoDB, Inc, [Võrgumaterjal]. Available: https://docs.mongodb.com/manual/core/document/. [Kasutatud 26 aprill 2019].

[26] „Cloud Storage - Online Data Storage | Cloud Storage,“ Google Inc., [Võrgumaterjal]. Available: https://cloud.google.com/storage/. [Kasutatud 26 aprill 2019].

[27] „Cloud Storage pricing | Cloud Storage | Google Cloud,“ Google Inc., [Võrgumaterjal]. Available: https://cloud.google.com/storage/pricing#network-pricing. [Kasutatud 26 aprill 2019].

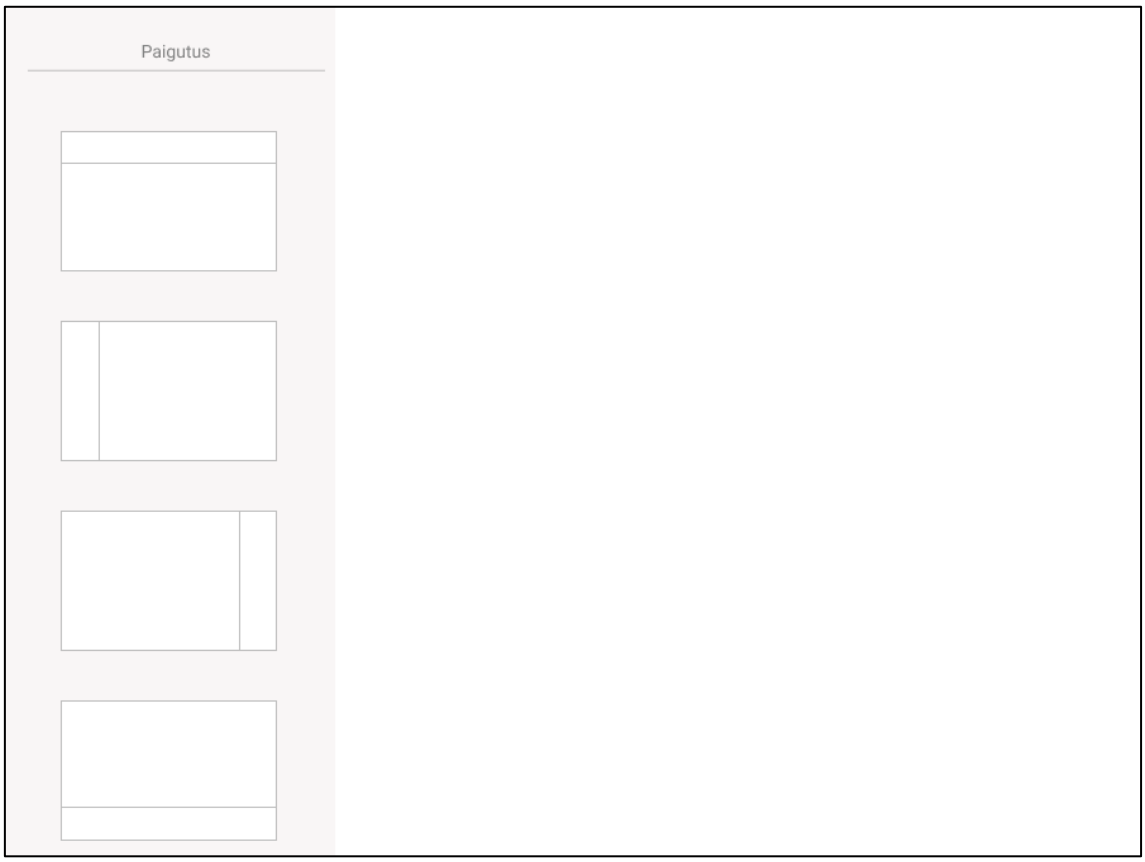
[28] „Enterprise Application Container Platform | Docker,“ Docker Inc., [Võrgumaterjal]. Available: https://www.docker.com/. [Kasutatud 5 mai 2019].

[29] „Production-Grade Container Orchestration - Kubernetes,“ The Linux Foundation, [Võrgumaterjal]. Available: https://kubernetes.io/. [Kasutatud 5 mai 2019].

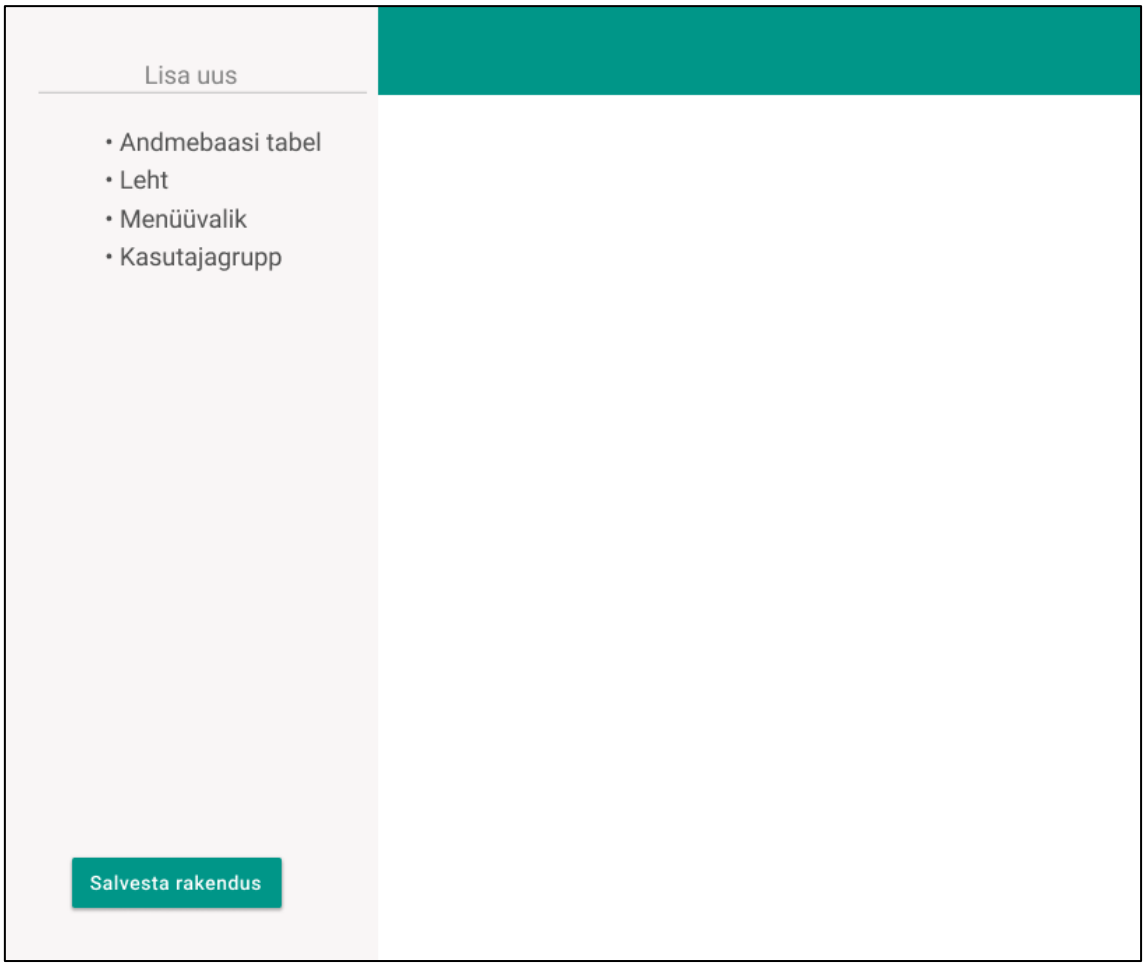
[30] „Avaliku sektori äriprotsessid. Protsessianalüüsi käsiraamat.,“ Majandus- ja kommunikatsiooniministeerium.

[31] H. Vallaste, „e-teatmik,“ [Võrgumaterjal]. Available: http://www.vallaste.ee/.

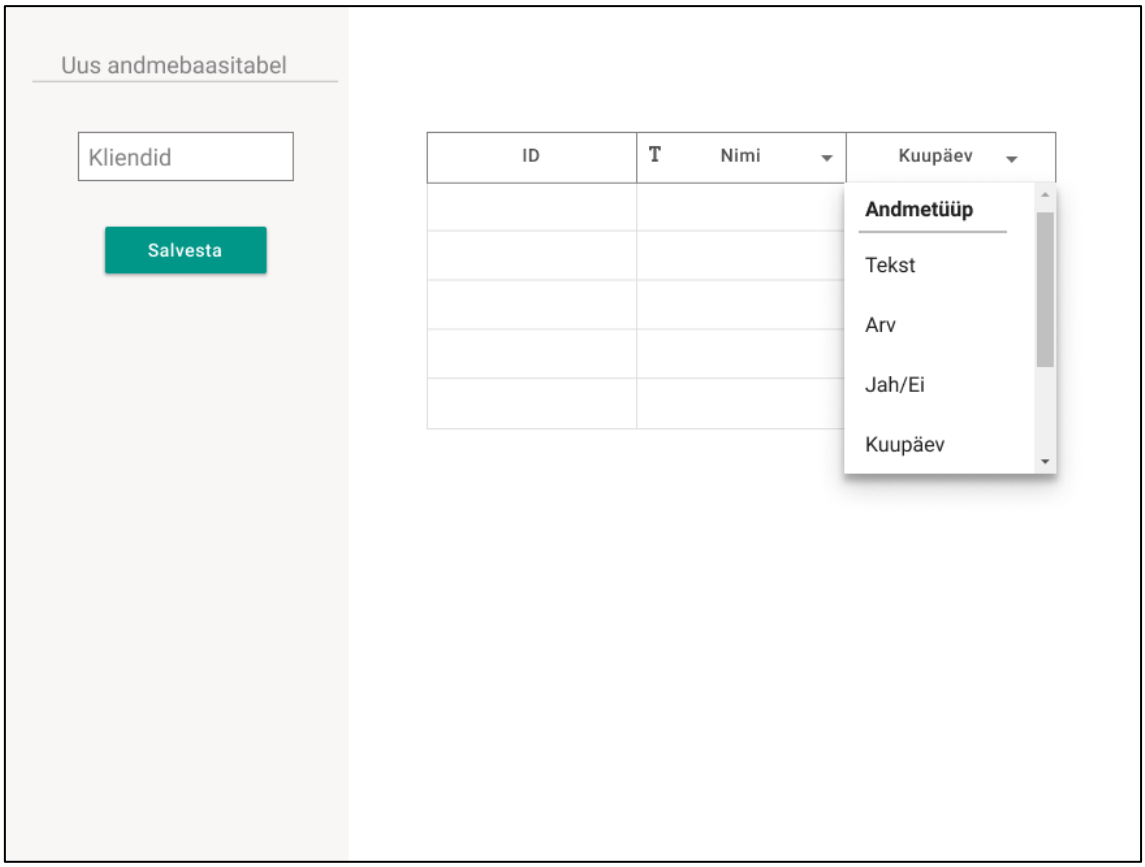
Lisa 1 – Esmase prototüübi vaated



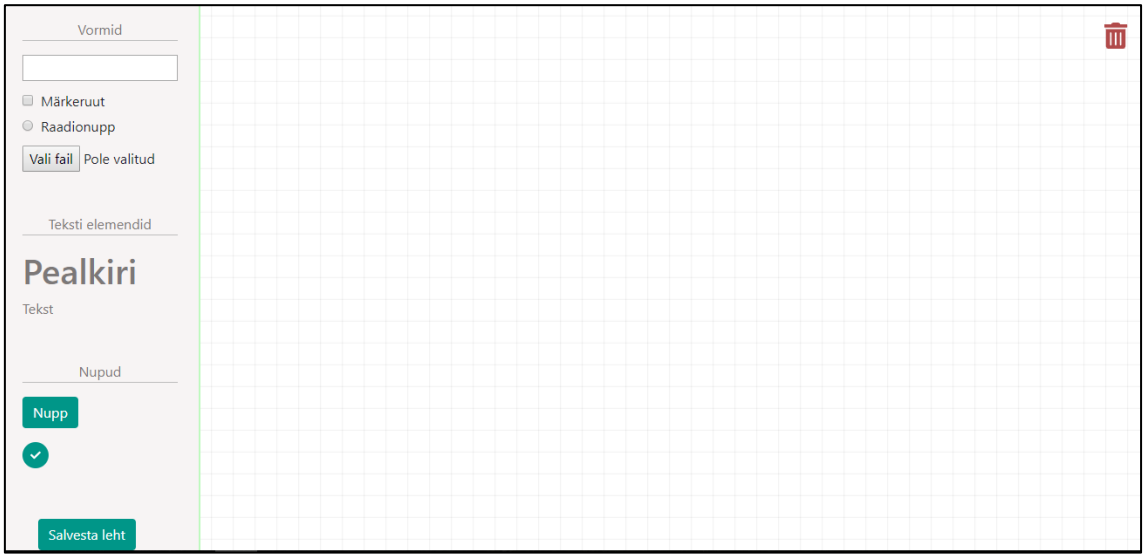
Joonis 20 Veebirakenduse üldise paigutuse valik



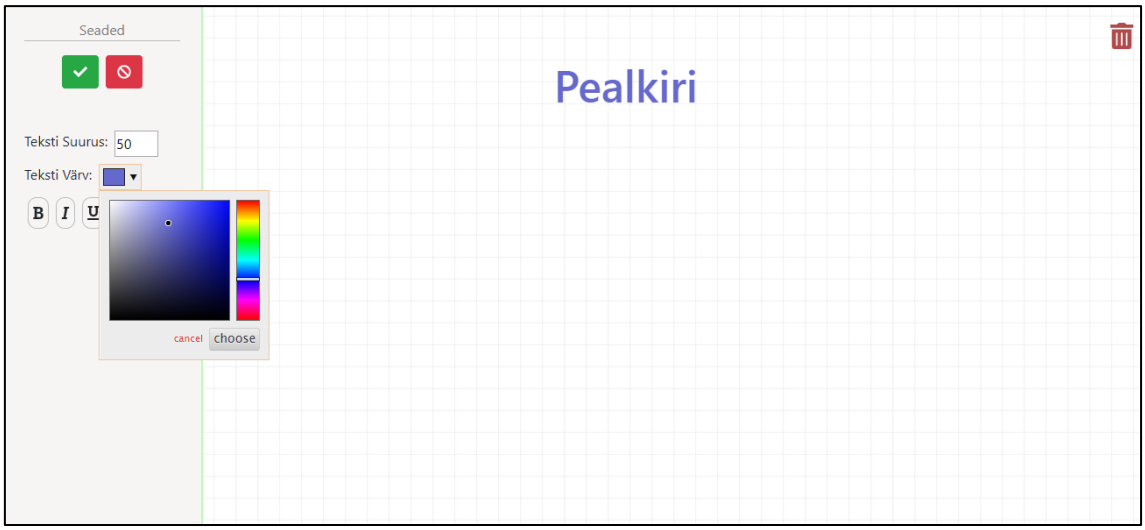
Joonis 21 - Rakenduse osade lisamise vaade



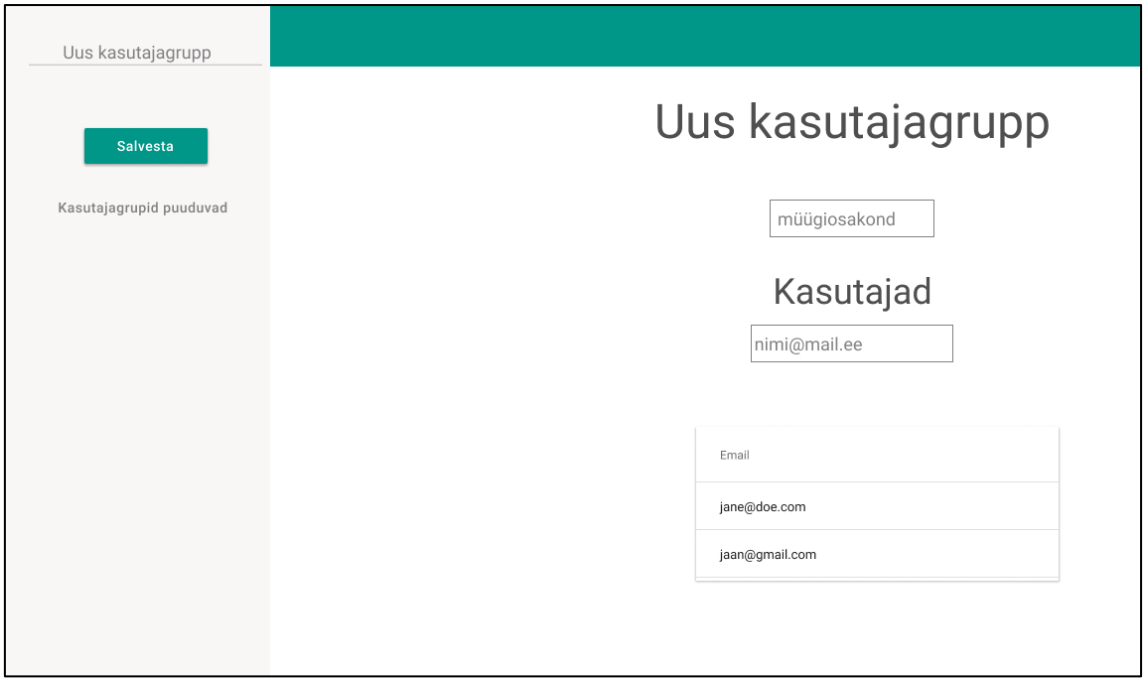
Joonis 22 - Andmebaasitabeli lisamise vaade



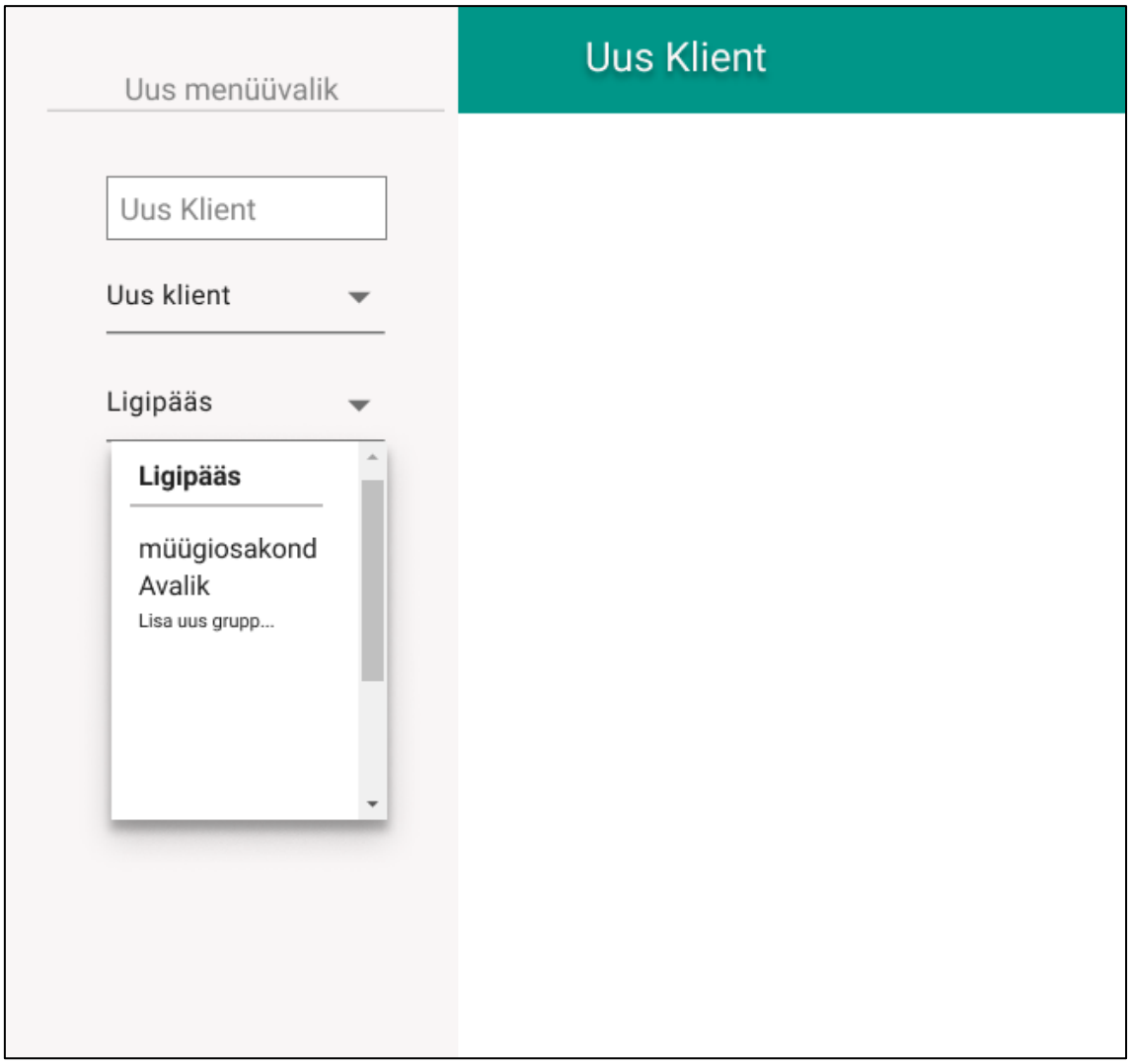
Joonis 23 - Uue lehe loomise vaade. Elemente saab lohistada sobivasse asukohta.



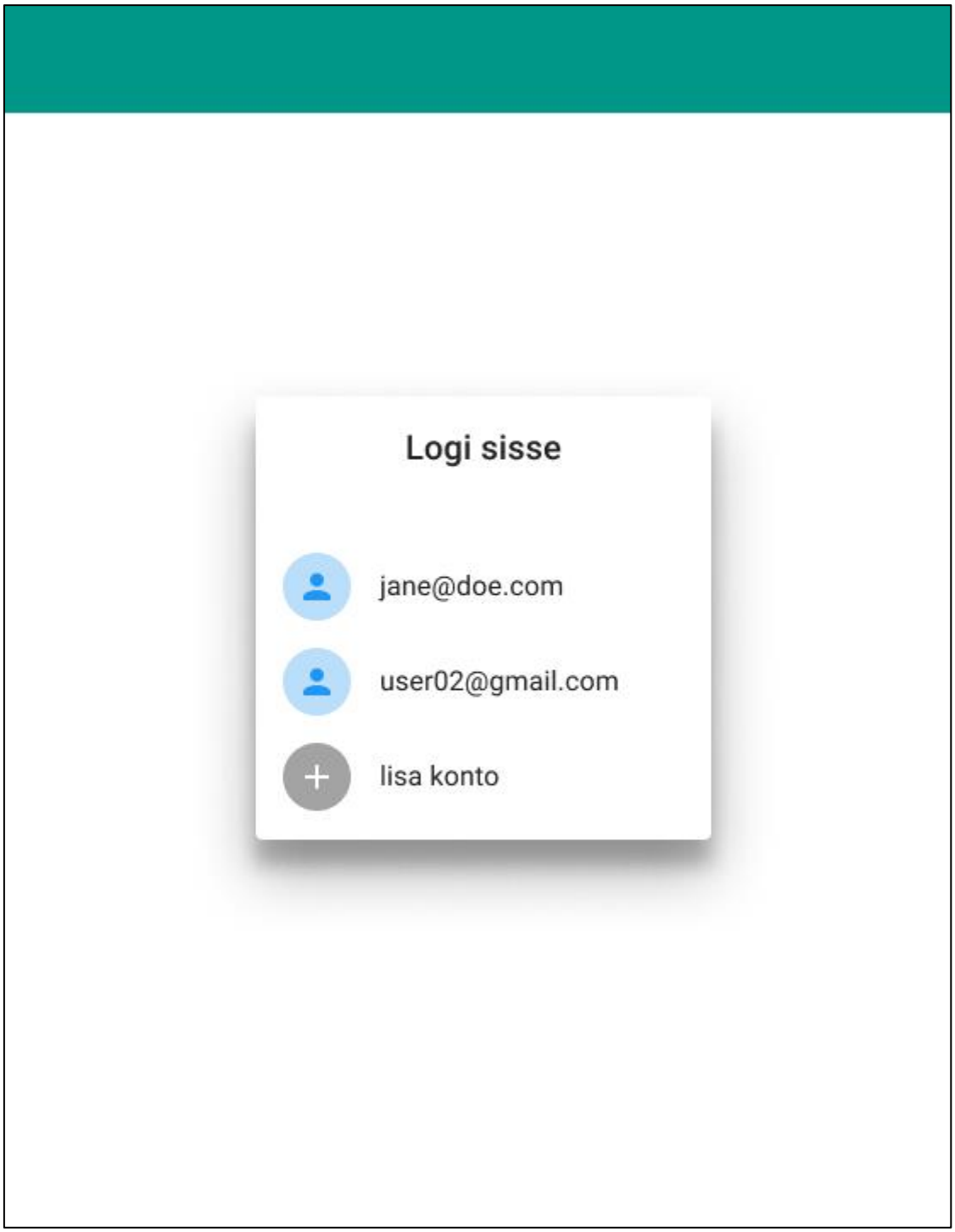
Joonis 24 - Lehe lisamise vaatel elemendi seadete muutmine



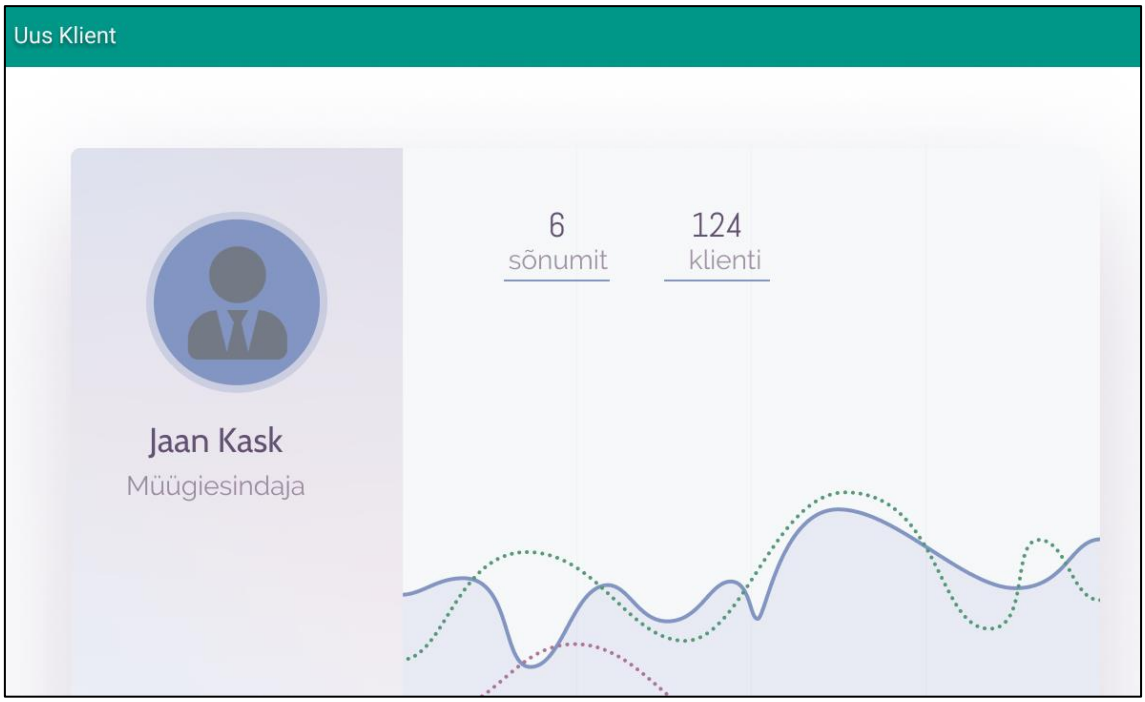
Joonis 25 - Kasutajagrupi lisamise vaade. Kasutajagrupid võimaldavad määrata ligipääsu erinevatele rakenduse osadele.



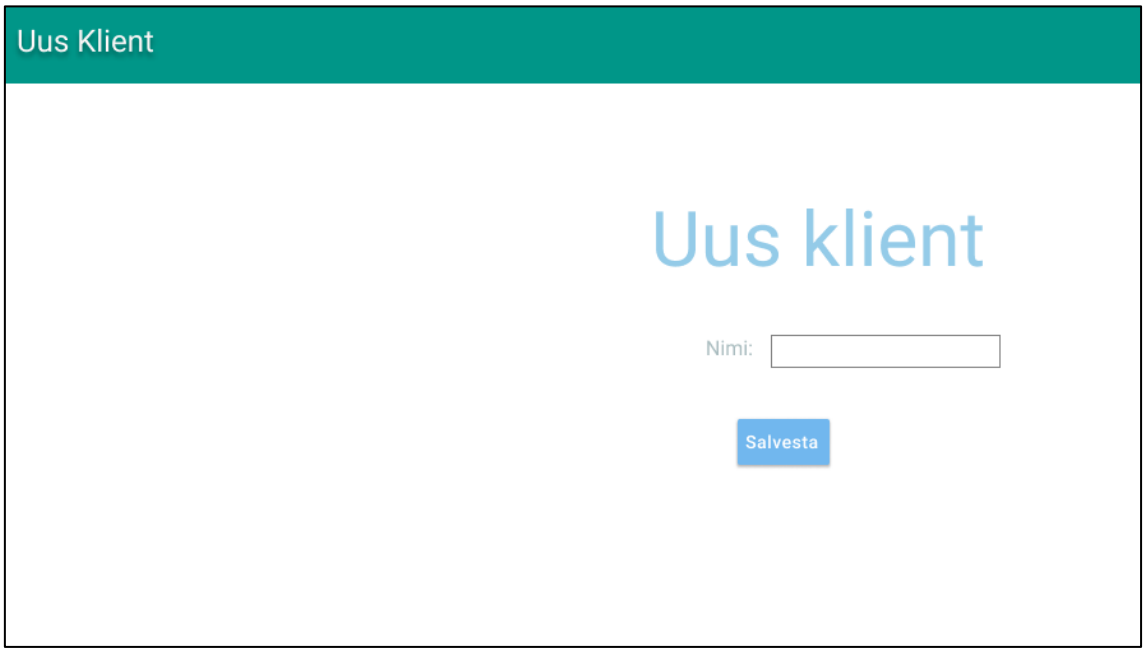
Joonis 26 - Menüüvaliku lisamine. Kasutaja saab määrata millisele lehele menüüvalik suunab ning kas sellel on ligipääsupiirang.



Joonis 27 - Valminud rakenduse vaade. Antud näidises on nõutud kasutaja autentimine.



Joonis 28 - Valminud rakenduse töölaua vaade. Prototüübis kuvatakse näidistöölauda, mida lõplik platvorm võimaldab samuti luua.



Joonis 29 - Eelnevalt loodud rakenduse lõppkasutajapoolne vaade

Lisa 2 - Kasutusmallid

Tabel 5. Kasutusmall 1 – kasutajaks registreerumine

ID ja nimetus	UC1 Kasutajaks registreerimine
Peamine aktor	Klient
Teisene aktor	-
Kirjeldus	Veebirakendusi luua soovivad kasutajad peavad olema end platvormil kasutajaks registreerinud. Kasutajaks on võimalik registreerida e-maili või enamlevinud autentimisteenuste kaudu (Facebook, Google).
Eeltingimused	Kasutajal ei ole selle e-posti aadressiga veel kasutajakontot.
Järeltingimused	Luuakse uus kasutajakonto, mille andmed salvestatakse platvormi andmebaasi.
Tavatöövoog	1. Kasutaja klõpsab nupul „Registreeru“. 2. Kasutaja valib registreerumise viisiks e-maili. 3. Süsteem valideerib sisestatud e-maili aadress. 4. Süsteem genereerib ja saadab kasutaja e-posti aadressile piiratud kehtivusajaga lingi, millelt avaneb parooli määramise vorm. 5. Kasutaja logib sisse enda postkasti ning klõpsab saadetud lingil. 6. Süsteem kontrollib kas avatud URL on veel kehtiv ning kehtivuse korral kuvab kasutajale parooli määramise vormi. 7. Kasutaja määrab endale parooli ning sisestab selle ka teistkordselt. 8. Kasutaja salvestab parooli. 9. Süsteem salvestab parooli krüpteerituna andmebaasi. 10. Süsteem annab kasutajale tagasiside, et parooli salvestamine õnnestus. 11. Süsteem suunab kasutaja sisselogimise vormile.
Alternatiivsed töövood	▪ Kasutaja valib registreerumiseks mõne autentimisteenuse. Süsteem ei saada e-postile parooli määramise linki ning eduka autentimise korral registreeritakse kasutaja. ▪ Kasutaja klõpsab parooli määramise lingil pärast selle kehtivuse lõppu. Süsteem teavitab sellest kasutajat eraldi vormil ning soovitab registreerumise protsess uuesti läbida.
Kasutussagedus	Ühekordne

Muu informatsioon	-

Tabel 6. Kasutusmall 2 – uue veebirakenduse lisamine

ID ja nimetus	UC2 Uue veebirakenduse lisamine
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Veebirakendusi luua soovivad kasutajad saavad lisada uue veebirakenduse.
Eeltingimused	Kasutaja on registreeritud ning sisse logitud.
Järeltingimused	Genereeritakse uus veebirakendus, mille andmed salvestatakse veebirakenduse noSQL andmebaasi.
Tavatöövoog	1. Kasutaja klõpsab nupul „Uus rakendus“. 2. Süsteem genereerib uue veebirakenduse põhja ning paneb valmis konteineri ja loob andmebaasiühenduse. 3. Kasutaja valib veebirakenduse üldise paigutuse. 4. Kasutaja salvestab rakenduse. 5. Süsteem salvestab veebirakenduse andmed veebirakenduse noSQL andmebaasi ning lisab viited platvormi andmebaasi ja seob need kliendiga.
Alternatiivsed töövood	▪ Kasutaja ei salvesta veebirakendust. Süsteem kustutab andmebaasi ja rakenduse konteineri.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 7. Kasutusmall 3 – uue tabeli lisamine veebirakendusele

ID ja nimetus	UC3 Uue tabeli lisamine veebirakendusele
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Kliendid saavad soovi korral kirjeldada ning luua veebirakenduses andmebaasitabeleid. Tabelite loomine ja andmete lisamine peab olema sarnane tabeltöötlustarkvaradega.
Eeltingimused	Veebirakendus on loodud.
Järeltingimused	Veebirakenduse andmebaasi on loodud uus tabel, mida on võimalik rakenduse töös kasutada.
Tavatöövoog	1. Kasutaja klõpsab lingil „Andmebaasi tabel“.

	2. Süsteem genereerib uue tabeli rakenduse noSQL andmebaasi. 3. Kasutaja määrab andmebaasitabelile nime. 4. Kasutaja valib igale veerule andmetüübi. 5. Kasutaja lisab soovi korral andmebaasi andmed. 6. Süsteem salvestab veebirakenduse andmed veebirakenduse noSQL andmebaasi ning lisab viited platvormi andmebaasi ja seob need kliendiga.
Alternatiivsed töövood	▪ Kasutaja ei määra tabelile nime. Süsteem kuvab kasutajale hoiatust.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 8. Kasutusmall 4 – uue vaate lisamine

ID ja nimetus	UC4 Uue vaate lisamine veebirakendusele
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Kliendid saavad lisada veebirakenduses vaateid (lehti). Vaadete lisamisel saab kasutada visuaalset programmeerimist ning valmis veebikomponente. Komponente saab paigutada vabalt valitud asukohta ning siduda andmebaasi tabelites olevate andmetega.
Eeltingimused	Veebirakendus on loodud.
Järeltingimused	Veebirakenduse andmebaasi on lisatud uus vaade.
Tavatöövoog	1. Kasutaja klõpsab lingil „Lisa uus leht“. 2. Süsteem kuvab uue lehe loomise vormi. 3. Kasutaja valib valmiskomponentide nimekirjast elemendi ning paigutab selle ekraanil sobivale kohale. 4. Kasutaja teeb muudatusi elemendi seadetes. 5. Kasutaja määrab vaatele nime. 6. Kasutaja salvestab vaate. 7. Süsteem salvestab vaate andmed veebirakenduse noSQL andmebaasi.
Alternatiivsed töövood	▪ Kasutaja ei määra vaatele nime. Süsteem kuvab kasutajale hoiatust.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 9. Kasutusmall 5 – uue kasutajagrupi lisamine.

ID ja nimetus	UC5 Uue kasutajagrupi lisamine veebirakendusele
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Kliendid saavad lisada veebirakenduses ligipääsude haldamiseks kasutajagruppe. Kasutajagruppi lisatakse kasutajad e-maili aadressi alusel. Ühes rakenduses võib olla mitu kasutajagruppi, et määrata lõppkasutajatele erinevaid ligipääsusid.
Eeltingimused	Veebirakendus on loodud.
Järeltingimused	Veebirakenduse andmebaasi on lisatud uus kasutajagrupp.
Tavatöövoog	1. Kasutaja klõpsab lingil „Lisa uus kasutajagrupp“. 2. Süsteem kuvab kasutajale uue kasutajagrupi loomise vormi. 3. Kasutaja sisestab grupi nime. 4. Kasutaja sisestab gruppi kuuluvate liikmete e-posti aadressid. 5. Kasutaja salvestab grupi. 6. Süsteem salvestab grupi andmed veebirakenduse noSQL andmebaasi.
Alternatiivsed töövood	▪ Kasutaja ei määra vaatele nime. Süsteem kuvab kasutajale hoiatust. ▪ Kasutaja ei sisesta ühtegi gruppi ühtegi liiget. Süsteem salvestab grupi ilma liikmeteta. Grupi nime on võimalik muuta ning hiljem kasutajaid juurde lisada või neid eemaldada.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 10. Kasutusmall 6 – uue menüüvaliku lisamine.

ID ja nimetus	UC6 Uue menüüvaliku lisamine veebirakendusele
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Kliendid saavad lisada loodavale veebirakendusele menüüvalikuid, et navigeerida erinevate vaadete vahel. Menüü asukoht on veebirakenduse loomisel kasutaja poolt määratud. Kasutaja saab määrata menüüvalikutele ligipääsupiiranguid.
Eeltingimused	Veebirakendus on loodud.
Järeltingimused	Veebirakenduse andmebaasi on lisatud uue menüüvaliku andmed.

Tavatöövoog	1. Kasutaja klõpsab lingil „Lisa uus menüüvalik“. 2. Süsteem kuvab kasutajale uue menüüvaliku loomise vormi. 3. Kasutaja sisestab menüüvaliku nime. 4. Kasutaja valib lehe, millele menüüvalik suunab. 5. Kasutaja salvestab menüüvaliku. 6. Süsteem salvestab menüüvaliku andmed veebirakenduse noSQL andmebaasi.
Alternatiivsed töövood	▪ Kasutaja ei määra menüüvalikule nime. Süsteem kuvab kasutajale hoiatust. ▪ Kasutaja valib lisaks suunatavale lehele ka kasutajagrupi, millel on menüüvalikule ligipääs. Süsteem salvestab andmebaasi lisaks suunatava lehe andmetele ka viite kasutajagrupile, millel antud menüüvalikule ligipääs on.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 11. Kasutusmall 7 – veebirakenduse kasutamine.

ID ja nimetus	UC7 Loodud veebirakenduse kasutamine
Peamine aktor	Lõppkasutaja
Teisene aktor	-
Kirjeldus	Lõppkasutaja kasutab kliendi poolt loodud veebirakendust.
Eeltingimused	Veebirakendus on loodud ja kasutajal on rakendusele ligipääs.
Järeltingimused	-
Tavatöövoog	1. Lõppkasutaja navigeerib veebilehitsejaga veebirakenduse aadressile. 2. Süsteem saadab päringu platvormi andmebaasi vastava aadressiga rakenduse leidmiseks. 3. Süsteem saadab päringu rakenduse noSQL andmebaasi veebirakenduse andmete leidmiseks. 4. Süsteem genereerib vastavalt saadud andmetele lõppkasutaja veebilehitsejas veebirakenduse.
Alternatiivsed töövood	▪ Lõppkasutaja poolt sisestatud aadressiga veebirakendust ei leitud. Süsteem saadab lõppkasutaja veebilehitsejale vastuse HTTP veakoodiga 404 (lehte ei leitud). ▪ Lõppkasutajal ei ole ligipääsu valitud veebirakendusele. Süsteem kuvab sisselogimise vormi.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Tabel 12. Kasutusmall 8 – klientide andmete vaatamine.

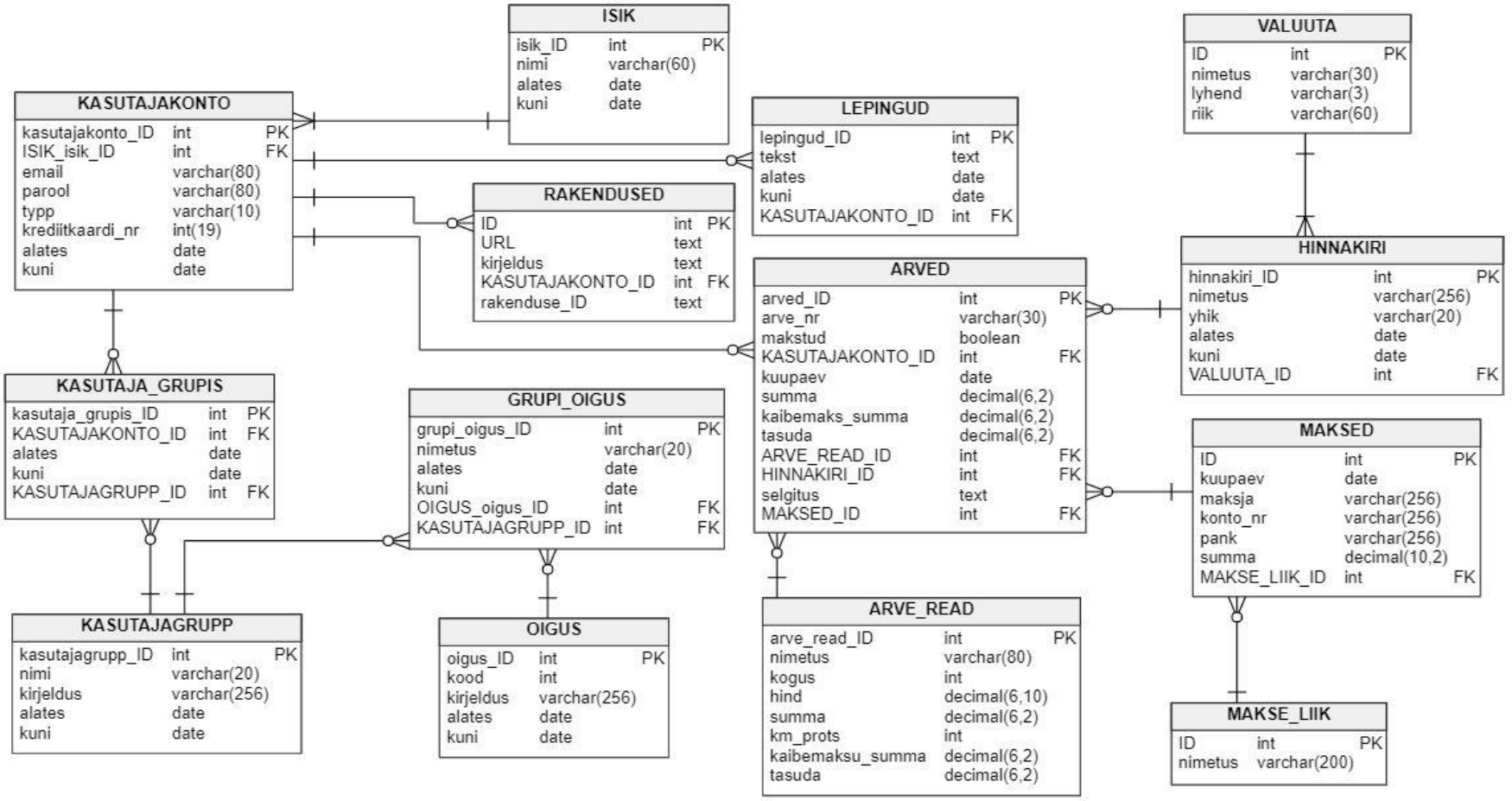
ID ja nimetus	UC8 Klientide andmete vaatamine
Peamine aktor	Administraator
Teisene aktor	-
Kirjeldus	Administraatori rollis kasutaja saab vaadata klientide andmeid. Klientide tabelit on võimalik filtreerida ning vaadata ühe kliendi detailvaadet.
Eeltingimused	Kasutaja on süsteemi sisse logitud administraatori rollis.
Tavatöövoog	1. Administraatori rollis kasutaja klõpsab lingil „Kliendid“. 2. Süsteem saadab päringu platvormi andmebaasi klientide andmete leidmiseks. 3. Süsteem kuvab klientide andmed (nimi, leping, kontaktid). 4. Administraator rollis kasutaja määrab tulemuste tabelis filtrid tulemuste filtreerimiseks. 5. Süsteem filtreerib välja klientide andmed vastavalt seatud filtritele ja kuvab filtreeritud andmetega tabelit. Eraldi uut andmebaasipäringut ei tehta.
Alternatiivsed töövood	▪ Andmebaasi ei ole sisestatud ühegi kliendi andmed. Süsteem kuvab kasutajale teavitust klientide puudumise kohta. ▪ Filtreerimise tulemusel ei leitud ühtegi tulemust. Süsteem kuvab kasutajale teavitust tulemuste puudumise kohta.
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	Klientide andmete vaatamiseks peab konkreetsel kasutajal olema vastav õigus.

Tabel 13. Kasutusmall 8 – veebirakenduse serveri ja andmebaasi haldamine.

ID ja nimetus	UC8 Veebirakenduse serveri instantside ja andmebaasi mahu ning asukoha muutmine
Peamine aktor	Klient
Teisene aktor	Administraator
Kirjeldus	Klient või administraatori rollis kasutaja saab vajadusel vaadata loodud veebirakenduse andmeid ja monitooringut ning muuta serverite arvu, andmebaasi mahtu ja nende asukohta.
Eeltingimused	Veebirakendus on loodud ja kasutajal on rakendusele ligipääs.
Järeltingimused	Veebirakenduse konteinerite ja andmebaasi andmed on muudetud.

Tavatöövoog	1. Kasutaja avab veebirakenduse seadete vaate. 2. Süsteem saadab päringu platvormi andmebaasi rakenduse andmete leidmiseks. 3. Süsteem kuvab ekraanile serveri monitooringu ning üldised andmed. 4. Kasutaja klõpsab seadete muutmise vormil. 5. Süsteem kuvab vormil muudetavaid seadeid. 6. Kasutaja muudab veebirakenduse infrastruktuuri seadeid ja salvestab vormi. 7. Süsteem arvutab uue hinna ning küsib kasutajalt kinnitust. 8. Kasutaja kinnitab uue hinnapaketi. 9. Süsteem salvestab rakenduse uued andmed platvormi andmebaasi. 10. Süsteem seab üles uued konteinerid ning muudab vajadusel andmebaasi instantsi.
Alternatiivsed töövood	-
Kasutussagedus	Vastavalt vajadusele
Muu informatsioon	-

Lisa 3 – Platvormi andmebaasi füüsiline andmemudel



Joonis 30. Veebiplatvormi andmebaasi andmemudel (Autori koostatud).

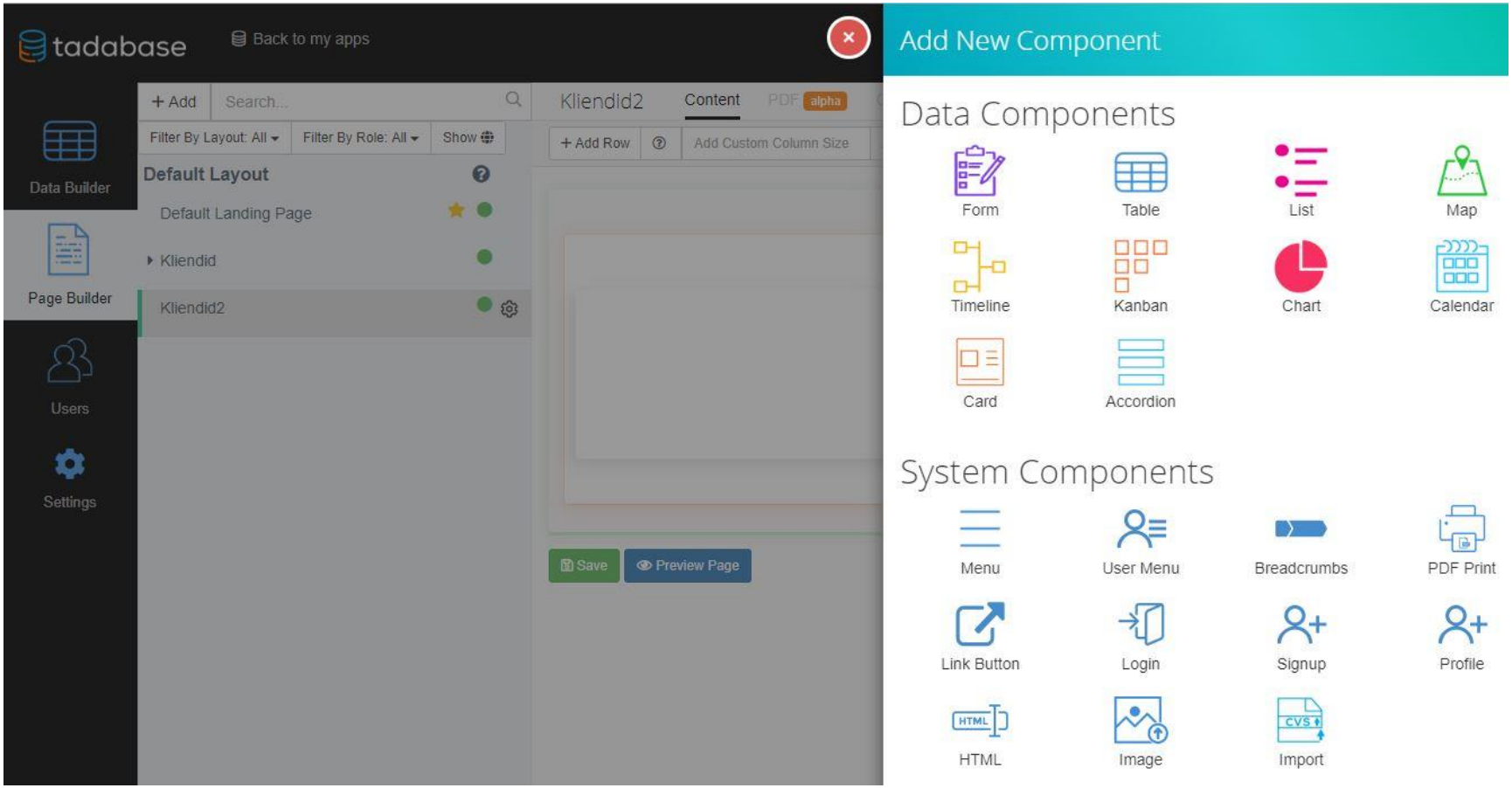
Lisa 4 – Relatsioonilise andmebaasi tabelite semantika

Tabel 14. Relatsioonilise andmebaasi andmemudeli tabelite semantika (autori koostatud).

Tabeli nimetus	**Semantika**
ISIK	Selles tabelis hoitakse kõigi isikute üldandmeid. Kuna ühel isikul võib olla mitu kasutajakontot hoiustatakse selles tabelis andmeid isiku enda, mitte kasutajakontode lõikes.
KASUTAJAKONTO	Kasutajakontode info. Erinevalt tabelist ISIK on siin lisaks veel e-posti aadress, räsitud parool, krediitkaardi number jne. Süsteemi esimeses versioonis kasutatakse autentimiseks kasutajanimena e-posti aadressi.
KASUTAJAGRUPP	Tabel kirjeldab kasutajate gruppe, kuhu kuuluvad kasutajad omavad sarnaseid õiguseid. Peamised grupid on näiteks „administraator“ ja „kasutaja“ aga ka erinevad süsteemi haldaja poolt loodavad grupid platvormi poolsetele esindajatele.
KASUTAJA_GRUPIS	Tabelis on andmed registreerunud kasutaja ja talle määratud kasutajagrupi kohta. Kasutaja grupi muutmisel ei muudeta mitte olemasolevat kirjet, vaid lõpetatakse eelnev kirje ja alustatakse uut ("alates", "kuni" atribuudid)
OIGUS	Kirjeldab erinevaid õiguseid, mida kasutajad võivad süsteemis omada. Igal õigusel on kood, mille alusel seda süsteemis identifitseeritakse.
GRUPI_OIGUS	Ühes kasutajagrupis olevate isikute õigused.
RAKENDUSED	Kasutaja poolt loodud rakendused. Tabelis seostatakse ära kasutaja ja rakendus. Lisaks rakendus ja URL. Selle tabeli alusel on võimalik lõppkasutaja suunata õigele rakendusele või kuvada rakenduse loojale tema rakendusi.
LEPINGUD	Kasutajatega sõlmitud lepingute info.
ARVED	Tabelis kirjeldatakse kõik üksikud arved, mis on kasutajatele esitatud.
ARVE_READ	Tabelis on kirjed kõikide arveridade kohta, mida kasutajatele esitatakse.
MAKSED	Tabel kajastab infot makstud arvete kohta. Kuna maksja ei pruugi olla registreeritud kasutaja on see eraldi atribuut ning võetakse kontoväljavõttelt või sisestatakse käsitsi.
MAKSE_LIIK	Tabel kirjeldab erinevaid makseliike, näiteks kaardimakse, pangaülekanne jne.

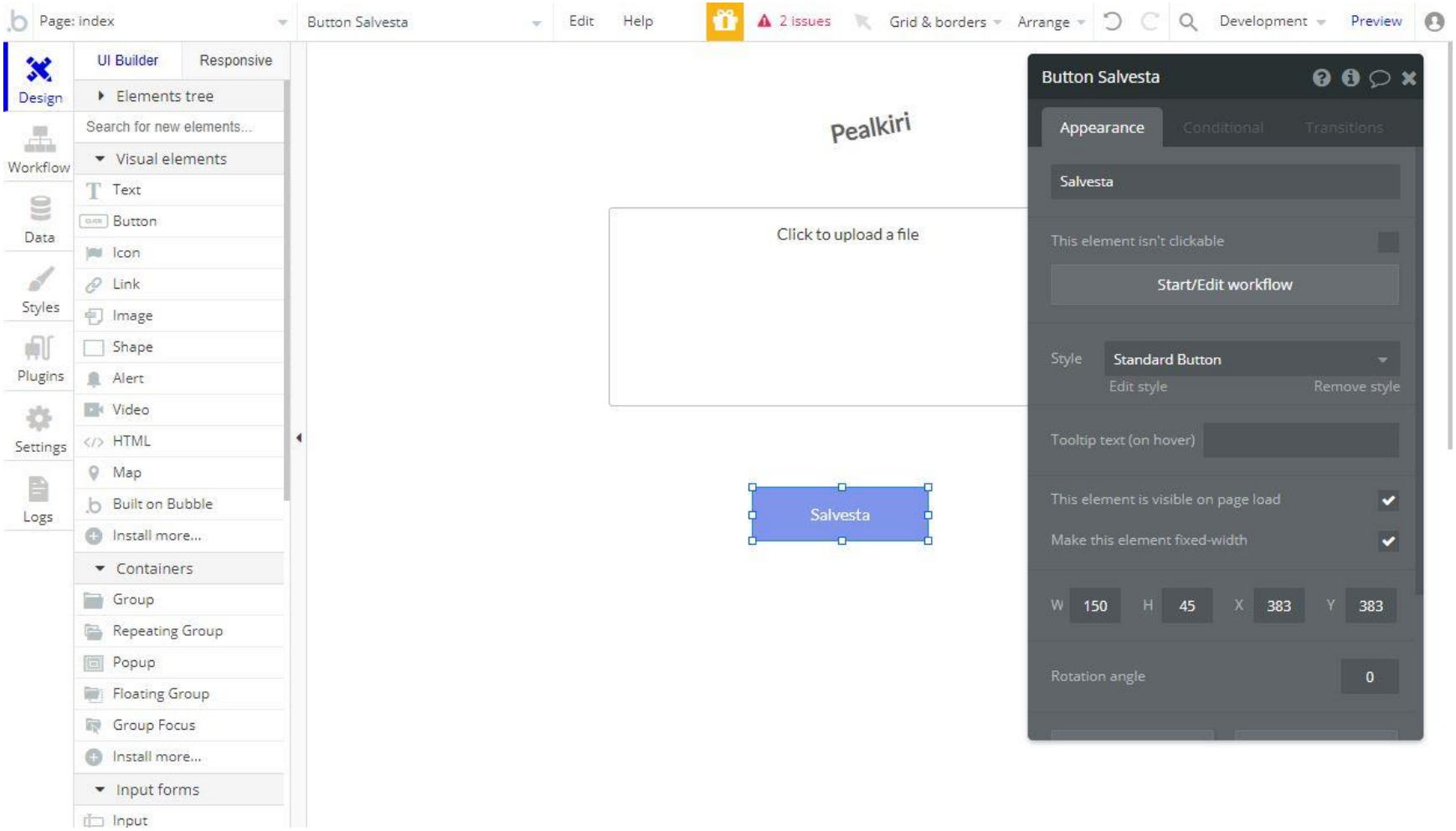
Tabeli nimetus	**Semantika**
HINNAKIRI	Tabelis on nimekiri teenuste hindadest erinevates valuutades.
VALUUTA	Tabelis on nimekiri erinevatest valuutadest, mida kasutatakse hindade juures.

Lisa 5 – Tadabase.io kasutajaliides



Joonis 31. Tadabase.io kasutajaliides (Allikas: autori koostatud)

Lisa 6 – Bubble.is kasutajaliides



Joonis 32. Bubble.is kasutajaliides (Allikas: autori koostatud)