TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Stefan Lill 213320IADB

Välireklaamide haldamissüsteemi kasutajaliidese loomine

Bakalaureusetöö

Juhendaja: Meelis Antoi

MSc

Tallinn 2025

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: Karl Stefan Lill

06.01.2025

Annotatsioon

Välireklaamid on laialdaselt kasutusel olev turundusstrateegia, mis on avalikus ruumis erinevatel viisidel toodete ja teenuste reklaamimine. Lõputöö raames esitatakse kasutajaliides, mis on osa välireklaamilahendusi pakkuva ettevõtte broneeringute haldamise süsteemist.

Lõputöö eesmärk on luua haldussüsteemile kasutajaliides, mis on kooskõlas ettevõtte poolt seatud nõuetega ning seeläbi aitab efektiivselt asendada hetkel kasutusel olevat piirangutega lahendust.

Töö põhiosa käsitleb probleemi püstitust, olemasolevate lahenduste analüüsi, planeeritava lahenduse analüüsi, ettevõtte näitel koostatavat süsteemi- ja ärianalüüsi, praktilise osa kirjeldust ning töö tulemuse hindamist, mille alla läheb ka võimaliku edaspidise arendusplaani kirjeldus.

Tulemusena valmib Vue.js raamistikus loodud kasutajaliides, mis arvestab ettevõtte poolt esitatud nõuete ja aktuaalsete veebidisainipõhimõtetega ning on lihtsasti integreeritav olemasoleva tagarakendusega. Tulemuse hindamise põhjaks võetakse ettevõttepoolsete kasutajate tagasiside loodud lahenduse kasutajasõbralikkuse kohta ja kooskõla ettevõtte esitatud nõuetega. Töös püstitatud eesmärk saavutatakse, kui ettevõtte tagasiside põhjal on võimalik järeldada, et lahendus võib edukalt asendada kasutusel oleva tööprotsessi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 6 peatükki, 9 joonist, 1 tabelit.

Abstract

Creating a User Interface for an Outdoor Advertising Management Platform

Outdoor advertising is a widely used marketing strategy that involves promoting products and services in public spaces through various means. In this case, the focus is on billboards, public transportation vehicles, and outdoor public infrastructure, for example bus pavilions.

As part of this thesis, a user interface for an outdoor advertisement management system is suggested, analyzed and developed for a company offering services of outdoor advertisement. This will be part of a complete booking management system, which aims to replace the current solution used by the company.

The thesis aims to state and set constraints for the problem at hand, analyze existing solutions, analyze the suggested solution, put together a thorough systems and business analysis based on a company case study, description of the practical solution and an evaluation of the outcome of the work, including a description of a possible future development plan.

As a result, there will be a user interface created in Vue.js framework, which considers the requirements set by the company and current web design principles. The practical part of the thesis will be developed using the agile development cycle, as the focus of the work is creating a user-friendly interface, which needs continuous coordination with the end-user. As from a technical viewpoint, the user interface can seamlessly be integrated with the existing backend application. The evaluation of the result will be based on the feedback of the business users on the user-friendliness of the solution and its compliance with the business requirements. The objective of the work will be achieved if the company's feedback can be used to conclude that the solution can successfully replace the existing workflow.

The thesis is in Estonian and contains 29 pages of text, 6 chapters, 9 figures, 1 table.

Lühendite ja mõistete sõnastik

ADA	*Americans with Disabilities Act*, 1990. aastal loodud tsiviilõigus, mis keelab diskrimineerida tulenevalt puudest
andmepood	ingl (Pinia) *store*, rakenduse seisundi haldamiseks mõeldud teek
API	*Application Programming Interface*, rakendusliides
CDD	*Component-Driven Development*, komponendi-põhine arendus
DOOH	*Digital out-of-home*, digitaalsed välireklaamid
eepos	ingl *epic*, suuremahuline kasutajalugu
edenemisriba	ingl *completion bar*, element progressi visualiseerimiseks
IDE	*Integrated Development Environment*, integreeritud arenduskeskkond
JSON	*JavaScript Object Notation*, veebirakendustes kasutatav andmevahetusformaat
JWT	*JSON Web Token*, JSON-vormingus pääsutalong
kapseldunud	ingl *nested*, hierarhiline struktuur, kus üks element või objekt asub teise sees
klient-server	ingl *client-server*, arhitektuurimudel, kus klient (tavaliselt rakendus või kasutaja) saadab päringuid ja server vastab nendele, pakkudes vajalikke andmeid või teenuseid
mikrointeraktsioonid	ingl *microinteractions*, väikesed kasutajakogemuse nüansid, mis aitavad anda kasutajale tagasisidet või suunata teda järgmise tegevuse juurde
oleku haldamine	ingl *state management*, teguviis, kuidas hallata rakenduse andmeid ja hetkeseisu
REST	*Representational State Transfer*, olekuta andmevahetusel põhinev veebirakenduste arhitektuuristiil
Routing	Veebirakenduses lehekülgede/komponentide vaheline navigeerimine, mida kasutatakse ühe-lehe rakenduste puhul
tüübikood	ingl *boilerplate code*, korduv koodiloogika/struktuur, mida esineb projekti põhifunktsionaalsuse seadistamisel
URL	*Uniform Resource Locator*, ühtne ressursilokaator
ühe-lehe rakendus	ingl s*ingle-page application*, arenduspõhimõte, kus veebilehel uuendatakse sisu dünaamiliselt, ilma tervet lehte värskendamata

WCAG	*Web Content Accessibility Guidelines*, standard veebilehtedel ligipääsetavuse tagamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lõputöös käsitletava probleemi aluseks on välireklaamidega tegeleva ettevõtte vajadus infosüsteemi järele, mis lihtsustaks ettevõtte jaoks kogu broneeringute haldamise tööprotsessi. Ettevõtte fookuses on prinditavad välireklaamid, mida paigaldatakse näiteks stendidele, plakatitele, bussipaviljonidele, ühistranspordile ja muudele sarnastele pindadele.

Hetkel kasutusel olev lahendus põhineb arvutustabelil, mis kasvava ettevõtte puhul muutub ühel hetkel piiravaks teguriks. Lisaks ei ole praeguse lahenduse komponentide vahel otseselt ühtegi kontrollivat seost, mis paneb broneeringuga seotud informatsiooni talletamise ja haldamise vastutuse täielikult ainult selle töötaja peale, kes broneeringuga algusest peale tegeles. See tähendab infokadu mõttes võrdlemisi suurt riski ning liigselt kohti tööprotsessis, kus infokadu võib tekkida.

Lahendus toob broneeringuhaldusega seotud aspektid tuua kokku ühte infosüsteemi, mis muudaks broneeringute haldamise efektiivsemaks ja vähendaks tööprotsessis kohti, kus infokadu võib tekkida. Antud töös pakutakse lahendusena välja infosüsteemi kasutajaliides. Tagarakenduse arenduse tegi oma lõputöös Jan Ulrich Sütt [1].

Töö eesmärgiks on pakutava lahenduse kohta koostada põhjalik analüüs ning selle põhjal arendada ettevõtte nõuetega kooskõlas olev kasutajaliides. Töö tulemusena valmiv kasutajaliides peab olema sujuvalt integreeritav olemasoleva tagarakendusega.

Töö põhiline sisu koosneb neljast suuremast peatükist, milleks on ülesandepüstitus, probleemi analüüs, kasutajaliidese arendus ning tulemust hindav osa. Probleemi analüüs koosneb nii üldistest veebidisainipõhimõtete analüüsist kui ka ettevõtte näitel tehtud ärianalüüsist. Kasutajaliidese arenduse peatükk koosneb lahenduse ja arendusprotsessi kirjeldusest, lisaks sellele ka dokumentatsiooni ja testimise kirjelduses. Tulemusi hindav peatükk kirjeldab töö autori ja ettevõtte hinnanguid lahendusele ning ka ettepanekuid edasiseks arendusplaaniks.

2 Ülesandepüstitus

Ülesandepüstituse peatükis seletatakse esmalt lahti probleem, mille lahendamist käsitleb lõputöö. Selle raames tutvustatakse lähemalt ka kasutusel olevat lahendust ning tuuakse välja selle puudujäägid ja piirangud. Järgmisena esitatakse ettevõtte ja töö autori poolt seatud lähtetingimused, mille põhjal tutvustatakse lühidalt pakutavat lahendust. Lõpetuseks loetletakse eesmärgid, mida pakutav lahendus peaks täitma ning metoodika, millega eesmärgipärase lahenduseni plaanitakse jõuda.

2.1 Probleemi kirjeldus

Lõputöö raames lahendatakse välireklaamindusega tegeleva ettevõtte probleem, kus keskendutakse ettevõtte tööloogika põhjal andmete haldamise ja tööprotsessi efektiivsemaks muutmisega. Järgnevalt seletatakse lahti lühidalt ettevõtte peamine tööprotsess ning tuuakse välja probleemsed kohad, mis praeguse lahenduse juures eksisteerivad.

Broneerimisprotsess algab kliendi poolt edastatava broneerimispäringuga, mis liigub kliendi ja ettevõtte vahel edasi-tagasi seni, kuni klient on valmis ettevõttega lepingut sõlmima või loobub teenusest. Lepingu sõlmimise puhul esitab välireklaami pakkuv ettevõte tellimuse trükikojale. Trükitellimuse täitmisel paigaldab välireklaami ettevõte lepingus sätestatud asukohtadesse kokku lepitud ajaks prinditud reklaamid.

Probleemne koht selle tööprotsessi juures on ettevõttesisene informatsiooni haldamine, kus praegune lahendus lubab liigseid riske info kadumises ning ettevõtte tulevikku arvestades on muutumas piiravaks teguriks. Pikemalt on lahti kirjutatud praegu kasutusel oleva lahenduse puudused järgmises peatükis. Loodav lahendus peaks ettevõttesisest infohaldust parandama, koondades kõik broneeringuga seotud informatsiooni ühte kohta, andes sellest tervikliku ülevaate ning eemaldama praeguse lahenduse poolt seatud piirangud ettevõtte kasvule.

2.2 Kasutusel olev lahendus

Hetkel ettevõttes kasutusel olev lahendus tugineb peamiselt Google Sheets'ile ja Google Drive'ile. Google Sheets'i kasutatakse kalendri ja muu broneeringuga seotud info haldamiseks, Google Drive'i kasutatakse broneeringuga seotud meediafailide hoiustamiseks ja trükikojaga jagamiseks. Selline lahendus on küll võimaldanud ettevõttel tööülesannete täitmisega toime tulla, kuid pole tulevikule mõeldes jätkusuutlik.

Peamiselt kerkib kasutusel oleva lahendusega esile kolm probleemi: pikk õppimiskõver, efektiivsus ja infokadu. Kokkuvõtvalt võib öelda, et selle lahenduse peamine probleem on kasutatavus. Pikk ajakulu lahenduse kasutama õppimiseks tähendab ettevõtte jaoks ärilist kahju, kui tekib vajadus välja koolitada uus töötaja. Mitmest komponendist koosnev praegune lahendus piirab ka tööprotsessi efektiivsust, sest vajaliku info leidmiseks ja lugemiseks tuleb seda otsida mitmest erinevast kohast. Infokadu põhiliseks põhjustajaks on fakt, et andmete sisestamine arvutustabelisse käib käsitsi ning ilma ühegi spetsiifilise kontrollita, mis peaks tegelikult broneeringute loomise või muutmise korral toimuma.

2.3 Lähtetingimused

Lõputöö eesmärkide saavutamiseks on vaja seada lähtetingimused, mis selgitavad ära töö skoobi ning annavad ette raamistiku, milles jõuda soovitud tulemuseni. Antud juhul on tegu kindla ettevõtte näitel arendatava kasutajaliidesega, seega selle disain peab eelkõige arvestama nende poolt esitatud nõuetega. Samas peab arvestama ka faktiga, et eduka tulemuse saavutamisel peab loodud infosüsteem olema kasutatav ka teiste samas valdkonnas tegutsevate ettevõtete poolt, mis seab omalt poolt piiranguid sellele, kui spetsiifiliseks on võimalik lõputöö raames valmiva kasutajaliidesega minna. Selle tagamiseks on vajalik üleüldiste disainipõhimõtete analüüs, et teostatav lahendus oleks võimalik oma kasutajamugavuse tõttu laialdaselt kasutusele võtta.

Töö praktilist poolt puudutavaid lähtetingimusi on kaks. Esiteks peab valitav raamistik, milles kasutajaliides kirjutatakse, olema paindlik ning lühikese õppimiskurviga, et rahuldava tulemuseni jõutaks piiratud aja sees. Teiseks peab olema valitav raamistik olema kergesti integreeritav olemasoleva tagarakendusega.

2.4 Eesmärk

Lõputöö peamine eesmärk on luua ettevõtte nõuetega kooskõlas olev kasutajaliides välireklaamide broneeringute haldamise infosüsteemi jaoks, mis aitaks asendada ettevõttes hetkel kasutusel olevat piirangutega tööprotsessi. Antud alapeatükis kirjeldatakse täpsemalt lahti ka analüütilise ja praktilise osa eesmärgid.

Analüütilisel osal on kolm eesmärki, millest esimene on analüüsida olemasolevaid lahendusi, tuues välja nende kitsaskohad, seeläbi põhjendades vajadust loodava infosüsteemi jaoks. Teine eesmärk on teha ettevõtte põhjal põhjalik äriärianalüüs ning tuua välja selle põhjal eripärad, millega peab arvestama, kui luuakse infosüsteemi välireklaamiga tegeleva ettevõtte jaoks. Kolmas eesmärk on analüüsida hetkel aktuaalseid veebidisaini põhimõtteid, luues taaskasutatavat väärtust, mida on võimalik eeskujuks võtta ka teistsuguste eesmärkidega infosüsteemide kasutajaliideste loomisel.

Praktilise osa eesmärk langeb kokku peamise eesmärgiga, milleks on tagarakendusega integreeritav kasutajaliides. Lisaks sellele peaks praktilise osa koostamise käigus valmima kasutajaliidese kohta ka adekvaatne dokumentatsioon ning kokkuvõte kasutajaliidese testimisest.

2.5 Metoodika

Lõputöö eesmärgi edukaks täitmiseks kavatseb autor kasutada järgmist metoodikat, mille esimesed kolm sammu lähevad analüütilise osa alla, neljas praktilise osa alla ning viimane tulemuste peatüki alla.

1. Nõuete kaardistamine ettevõttega, et selgeks teha ettevõtte soovid ning nende põhjal formuleerida esialgne arendusplaan.
2. Olemasolevate lahenduste analüüs, et tuua välja nende kitsaskohad ja leida asjad, millele uut lahendust luues peaks tähelepanu pöörama.
3. Teha ettevõtte näitel süsteemi- ja ärianalüüs, et panna paika konkreetne raamistik arendustööks.
4. Viia agiilse tsükli abil läbi analüüsile põhinev kasutajaliidese arendus.
5. Hinnata tulemusi autori vaatepunktist ning analüüsida ettevõttelt tulnud tagasisidet.

2.6 Pakutav lahendus

Lõputöös pakutav lahendus on kasutajaliidese osa terviklikust infosüsteemist. Probleemi otsustati kahes osas käsitleda, sest antud lähenemine tagab ajaliselt piiratud arendusperioodi jooksul põhjalikuma lähenemise tagarakendusele ja kasutajaliidesele. Seda nii analüüsis kui ka praktilises osas.

Kui infosüsteemi tagarakenduse arendusel loodi välireklaami spetsiifikaga arvestava broneeringute haldussüsteemi jaoks kriitiline infrastruktuur, siis kasutajaliidese arendusega tuleb tagada infosüsteemi kasutatavus. Lahenduse peavad saama probleemid, millest kirjutati peatükis 2.1. Idee on luua veebipõhine kasutajaliides, mis teeb broneeringuga seotud info talletamise ja hiljem ka haldamise loogiliseks ning intuitiivseks. Seeläbi saavad lahendatud hetkel kasutusel oleva lahenduse probleemid, milleks on pikk õppimiskõver ja efektiivsus. Vähendatakse ka kohti tööprotsessis, kus võib tekkida infokadu.

Eesmärgipärane lahendus oleks vähemalt Eestis unikaalne, sest hetkel ei ole avalikult saadaval ühtegi lahendust, mis oleks orienteeritud välireklaamiga tegelevatele ettevõtetele. Olemasolevate lahenduste, mis teoorias võiksid rahuldada ettevõtte vajadusi, kohta käiv analüüs on pikemalt lahti kirjutatud peatükis 3.1.

3 Probleemi analüüs

Töö analüütilise osa eesmärgiks on eelkõige põhjendada arendatava infosüsteemi vajalikkust ning välja tuua selle eripärad. Selleks on vajalik analüüsida olemasolevaid lahendusi, tuues välja nende tugevused ja nõrkused, millele võiks töö raames arendatavat infosüsteemi luues tähelepanu pöörata. Lisaks sellele analüüsitakse ka hetkel aktuaalseid veebidisainipõhimõtteid, et panna paika raamistik, milles kasutajaliidese arendus toimub.

3.1 Olemasolevate lahenduste analüüs

Olemasolevad lahendused jagunevad üldiselt kolme suuremasse kategooriasse, millest igal on omad eelised kui ka puudujäägid. Antud peatükis analüüsitakse igast kategooriast näitena mõnda lahendust ning tuuakse välja konkreetsed puudused, mis muudavad selle kategooria alla käiva lahenduse lõputööga aidatava ettevõtte jaoks mitte sobivaks.

3.1.1 Üldised lahendused

Üldiste lahenduste all on autor silmas pidanud sarnase funktsionaalsusega programme ja platvorme, nagu ettevõttes hetkel kasutatakse. Peamised probleemid selliste lahenduste juures on seotud platvormide üldsusega, mis muudab nende kasutamise välireklaamindusega seotud spetsiifika puhul liiga tülikaks.

Ühe väga üldise lahendusena on võimalik kasutada Google Calendar-i, mis küll broneeringute aja haldamiseks ning läbi Google Drive-i integratsiooni broneeringutega seotud failide säilitamiseks sobiks, ei arvesta välireklaaminduse spetsiifikaga. Kõige suurem puudujääk on välireklaamiasukohtasid puudutav aspekt. Iga asukoha jaoks oleks vaja luua eraldi kalender, mis muudaks näiteks kliendi päringu vastuse koostamise keeruliseks ning aeganõudvaks. Efektiivselt töötamiseks nõuaks see, et ettevõttes töötav inimene on korraga kursis mitmekümne erineva kalendriga, mis ei tundu väga reaalne. Lisaks sellele tõstaks see märgatavalt uue töötaja jaoks aega, et süsteemi toimimisest aru saada. Failide ning muu informatsiooni säilitamise koha pealt ei ole Google Calendar-is ühtegi validatsiooni, mis kontrolliks, et kõik vajalikud andmed on olemas ning korrektsed.

Üldiste lahenduste all võib välja tuua ka projektijuhtimiseks mõeldud platvormid nagu näiteks Asana. Asana on eeskätt mõeldud projektiga seotud ülesannete ja ressursside haldamiseks ning delegeerimiseks, eelisena on nad välja toonud platvormi skaleeritavuse ning sujuva kasutuskogemuse [2]. Lisaks nimetatud eelistele pakuvad nad palju võimalusi erineva tarkvara integreerimiseks [3], millega Asana integreerimine võiks teoorias minimaalselt rahulda ettevõtte vajadused. Sarnaselt Google Calendar-iga oleks kaetud ajahalduse ning failide säilitamise vajadused, kuid ilmnevad samad probleemid. Mitme reklaamiasukoha korraga haldamine muutuks liigselt keerukaks ning puuduvad ka broneeringutega seotud validatsioonid. Lisaks sellele tekib Asana näol ettevõttele uus püsikulu, mis ei pruugi ennast äriliselt õigustada. Erinevate integratsioonide lisamisel suureneks see veelgi. Viimase puudusena saab veel välja tuua sõltuvuse kolmanda osapoole tarkvarast, ning kuna tegu on suure kliendibaasiga platvormiga, ei saa probleemide esinemisel välireklaamiga tegelev ettevõtte tõenäoliselt piisavalt kiiresti tehnilist abi.

Kokkuvõtvalt võib öelda, et üldiste kalendripõhiste platvormide kasutamine võiks ettevõtte vajadused samal määral lahendada nagu nende praegu kasutusel olev lahendus seda teeb, kuid eksisteerivad probleemid nendega lahendatud ei saaks. Peamised kitsaskohad ilmnevad välireklaamindusega seotud spetsiifikale mõeldes, eriti mitme asukoha korraga haldamises ning broneeringuga seotud validatsioonide puudumises. Lisaks sellele oleks tasulised platvormid ettevõttele lisakulu ning ebavajalik sõltuvus kolmanda poole tarkvarast.

3.1.2 Turuplatsi tüüpi lahendused

Turuplatsi tüüpi lahenduste alla käivad platvormid, mis töötavad vahendamise põhimõttel. Tegu ei ole ettevõttepõhise haldustarkvaraga, vaid läbivaks jooneks on nn digitaalse turuplatsi loomine, kus on võimalik meediumite omanikel (nagu näiteks lõputööga aidataval ettevõttel) avalikustada oma pakutavaid reklaamiasukohti ning nende renditingimusi. Sellised platvormid on näiteks DoMedia [4], SignKick [5] ja VIIOH [6]. Kasutussõbralikkuse ja tehniliste aspektide poolest ei ole antud näidetele võimalik täpset hinnangut anda, sest puuduvad piisavad avalikud materjalid nende teemade analüüsimiseks. Küll aga äriidee kirjelduse järgi tõuseb esile mitu probleemi, mis lõputöö eesmärgis kirjeldatud lahendusega vastuolus on.

Kõige suurem probleem on asjaolu, et need platvormid ei lähe kokku äriprotsessiga, mis on kasutusel ettevõttes ning mida üritab lõputöö raames saavutatav tulemus täiendada. Turuplatsi tüüpi lahenduse kasutuselevõtuga kaoks ettevõttel võimalus kogu broneeringuprotsessi ise hallata. Lisaks ilmneks sama probleem nagu teiste ettevõtteväliste tarkvarade kasutuselevõtuga – sõltuvus kolmandast osapoolest, mis antud juhul on välditav. Pealegi ei ole Eestis hetkel teadaolevalt ühtegi sellist lahendust pakkuvat ettevõtet, mis on arvatavasti Eesti väikese turu tõttu. Suure tõenäosusega ei tasuks siia turule sisenemine turuplatsi tüüpi lahendust pakkuvatele ettevõtetele äriliselt ära.

3.1.3 Valdkonnaspetsiifilised lahendused

Antud kategooria alla käivad tarkvaralised lahendused, mis on mõeldud välireklaami pakkumisega tegelevatele ettevõtetele ning täidavad enamiku nõudmistest, mille on esitanud lõputööga aidatav ettevõte.

Üks sellistest teenusepakkujatest on ettevõtte BillboardPlanet, kelle välireklaamibroneeringute haldamiseks mõeldud tarkvara Quantum [7] täidab pealiskaudse analüüsi järel ära kõik aidatava ettevõtte nõudmised ära. Tarkvaras on olemas asukohtade ja ajakava haldus, paigaldusprotsessi jälgimise võimekus, raamatupidamise võimalus ning ka kliendihaldusega seotud funktsionaalsus. Kui eelnevalt analüüsitud lahendusetüüpide puhul olid selged äriprotsessiga seotud puudujäägid, siis Quantumi ja teiste sarnaste lahenduste puhul tekivad aidatava ettevõtte suurust ning vajadusi vaadates küsimused, et kas need tasuvad ennast finantsiliselt ära. Quantumi näitel saab välja tuua, et sarnastes tarkvarades on tihti DOOH (*Digital out-of-home)* võimekus, millest lõputöö lahendatava probleemi lahendamisel kasu ei ole. Arvestades, et sellised lahendused on arendatud suuremate turgude jaoks, kui seda on antud valdkonnas Eesti, siis võib nende kasutuselevõtt osutuda ettevõtte jaoks äriliselt kahjumlikuks. Kuna enamusel valdkonnaspetsiifilisest tarkvaradest puudub avalik hind, siis on sellele hetkel raske hinnangut anda.

Üldiselt ei ole Eesti turul selliseid lahendusi avalikult reklaamitud või analüüsitud, seega ei ole võimalik lõplikult öelda, kas selle kategooria alla käivad lahendused sobiks Eesti turule ning aidatavale ettevõttele. Lõputöö praktilise osa aspektist on mõistlik võtta arendusprotsessis eeskujuks selliseid lahendusi nagu on BillboardPlanet-i Quantum, aga

seejuures tuleb arvestada Eesti turu suuruse ning ettevõtte jaoks vajaliku funktsionaalsuse piiridega.

3.2 Disainipõhimõtete analüüs

Käesolevas alapeatükis käsitletakse erinevate kasutajaliidese osade disainipõhimõtteid. Seda tehakse üldises võtmes, et lõputööl oleks lisaks välireklaamindusega seotud spetsiifikale ka üldine taaskasutatav väärtus, mida oleks võimalik üleüldiselt kasutajaliidese arendusel kasutada. Täpsemalt analüüsitakse kasutajaliidese infrastruktuuri, mõistetava kasutusvoo ning mugava kasutajakogemuse loomist. Vaatluse all on nimetatud teemade all pigem hiljutisemad trendid, et tuua välja uudseid lähenemisi. Autor soovib ära mainida, et ajapiirangu tõttu ei implementeerita kõiki põhimõtteid, mida järgmistes alapeatükkides analüüsitakse.

3.2.1 Ligipääsetavuse tagamine

Veebirakenduste ligipääsetavuse tagamiseks on loodud tänapäeval erinevaid standardeid, millest tuleks arendustöid tehes lähtuda. Kõige tuntum neist, mille järgi ka spetsiifilisemad dokumentatsioonid on loodud, on WCAG (*Web Content Accessibility Guidelines*). WCAG on W3C (*World Wide Web Consortium*) poolt koostatud reeglite ja soovituste raamistik, mis aitab parandada veebirakenduse ligipääsetavust. Viimane WCAG versioon on WCAG 2.2, mis avaldati 2023. aasta oktoobris. Selles dokumendis räägitakse lähemalt veebirakenduse sisu tajutavusest, toimingute tegemisest, arusaadavusest ning sisu stabiilsusest. [8]

Tänu tehisintellekti arengule on loodud hiljuti erinevaid analüütilisi tööriistu, mis vastavalt WCAG või mõnele muule standardile on võimelised analüüsima veebirakenduste ligipääsetavust. Hea näitena võib välja tuua Siteimprove-i *Website accessibility checker*-i. Peale üldistele suunistele vastavuse analüüsimisele on võimalik ka kontrollida, et veebirakenduse ligipääsetavus oleks kooskõlas vastava regiooni seadustega, näiteks Ameerika Ühendriikides kehtestatud Ameerika Ühendriikide puuetega inimeste seaduse (ADA, *Americans with Disabilities Act*) ning rehabilitatsiooniseaduse paragrahviga 508 [9]. Ligipääsetavuse tagamist peaks innustama ka Google-i poolt kasutatava *Core Web Vitals* algoritmi kasutamine, mis tõstab parema ligipääsetavusega veebilehed otsingutulemustes ettepoole [10].

3.2.2 Mõistetava kasutusvoo loomine

Arusaadava kasutusvoo loomine on oluline, et kasutaja saavutaks veebirakenduse kasutamisel oma eesmärgid. Sarnaselt muudele valdkondadele, on ka veebidisaini puhul viimasel ajal olnud selge suund minimalistlike disainipõhimõtete poole. Üheks tähtsaks argumendiks selle puhul on kognitiivse koorma vähendamine, et vältida kasutaja segadusse ajamist [11] järgmise sammu suhtes. Esile on tõusnud ka mikrointeraktsioonide (ingl m*icrointeractions*) kasutamine ning seeläbi reaal-ajas kasutajale tagasiside andmine [12]. Sellega kinnitatakse kasutajale, et kas tema sooritatud tegevus oli edukas või kukkus läbi ning mis võis läbikukkumise põhjus olla. Järjepidevalt areneb ka isikupärastatud kasutajaliideste arendus, kus arvestatakse kasutaja eelmiseid rakendusesiseseid tegevusi ning muudetakse nende põhjal järgmisel kasutuskorral näiteks elementide või muu sarnase paigutust, et muuta kasutaja jaoks kasutusvoog efektiivsemaks. Üleüldiselt on rõhk selgetel ja lihtsatel kasutusteekondadel, mis aitavad kasutajal täita oma eesmärgid minimaalse vaevaga.

3.2.3 Mugava kasutajakogemuse loomine

Mugava kasutajakogemuse loomine tähendab üldist rahulolu, milleni kasutajad peaksid veebirakendust kasutades jõudma. Tähtsal kohal on veebirakenduse jõudlus, mis mõjutab nähtavalt kasutajakogemust. Näiteks kiired laadimisajad ning sujuvad üleminekud tegevuse sammude vahel on faktorid, mis panustavad positiivsesse kasutajakogemusse [13]. Tähelepanu tuleks pöörata ka mugavusfunktsioonidele, näiteks tumeda režiimi (ingl *dark mode*) olemasolule. Tumeda režiimi kasutamine aitab kasutajal näiteks vähendada ekraanist tulevat sinist valgust ning liigset kontrasti ümbritseva keskkonnaga (näiteks kui kasutaja on pimedamas keskkonnas). Mõlemad faktorid võivad silmi väsitada ning kasutajale ebamugavust tekitada [14], [15]. Paremaks kasutajakogemuseks jälgitakse ka visuaalse hierarhia põhimõtteid, kus tuuakse veebirakenduse elemendid näiteks suuruse, värvi, paigutuse või mõne muu silmapaistva omadusega välja [16]. Viimasena võib välja tuua ka näiteks mängulised elemendid erinevate protsesside juures. Mainitud elementide alla käivad näiteks protsessi progressi visualiseerivad edenemisribad (ingl *completion bars*), mis julgustavad kasutajad aktiivselt oma algatatud tegevusi lõpule viima [17].

3.3 Funktsionaalsete nõuete kirjeldus

Funktsionaalsed nõuded on nõuded, mis kehtestatakse infosüsteemi äriloogika kohta. Neid kirjeldatakse vastavalt kliendi esitatud äriprotsesside kirjeldustele. Üldiselt kirjeldavad funktsionaalsed nõuded ära infosüsteemi käitumise mingi teatud sisendi korral, olgu selleks näiteks andmete lisamine, muutmine, kustutamine, või mõni muu tegevus [18]. Arendatava infosüsteemi funktsionaalsed nõuded on paika pandud dialoogis aidatava ettevõttega, koostöös infosüsteemi tagarakenduse arendajaga. Funktsionaalsed nõuded on arusaadavuse eesmärgil kirjeldatud agiilsetele arendusmeetoditele (nagu näiteks *Scrum* ning *Kanban* [19]) omasele kes-mida-miks põhimõttele, mis kirjeldab tegevused järgmiselt:

- Kes – tegevuse sooritaja
- Mida – mis tegevust üritatakse parasjagu sooritada
- Miks – mis on tegevuse sooritamise eesmärk

Kirjeldatavaid rolle on infosüsteemis kolm: administraator, müügiagent ning paigaldaja. Kasutajaroll määrab ära, millistele andmetele ning tegevustele on kasutajal süsteemis ligipääs.

Kasutajalood on koondatud kuue eepose (ingl *epic*) alla, mis grupeerivad sarnase eesmärgiga kasutajalood. Lisaks kuuele spetsiifilisele eeposele on ka kõiki kasutajarolle puudutav kasutaja autentimise kasutajalugu, mis on igas eeposes, kuid kordamise vältimiseks ei ole seda iga eepose all eraldi välja toodud. Kasutajalugudel on numbrilised identifikaatorid, mis on iga eepose all unikaalsed. Eepos on kirjeldatud üldise tegevuse järgi, millega selle alla kuuluvad kasutajalood tegelevad:

1. Kasutajate haldamine – kasutajarolliks administraator, tegevused on kasutajate lisamine, kasutajadetailide vaatamine, kasutajate haldamine (kasutajalood 1-3)
2. Broneeringute haldamine – kasutajarolliks müügiagent, tegevused on broneeringute lisamine, broneerigu detailide vaatamine, broneeringute haldamine (kasutajalood 4-6)
3. Klientide haldamine – kasutajarolliks müügiagent, tegevused on kliendiprofiilide lisamine, kliendiprofiili detailide vaatamine, kliendiprofiilide haldamine (kasutajalood 7-9)

4. Reklaamobjektide haldamine – kasutajarolliks müügiagent, tegevusteks reklaamiasukohtadega, reklaampindadega ning reklaamiobjektidega seotud toimingud[1] (kasutajalood 10-18)
5. Reklaamkujunduste edastamine – kasutajarolliks on müügiagent, tegevuseks on reklaamkujunduse edastamine trükikojale (kasutajalugu 19)
6. Broneeringu täitmine – kasutajarolliks on reklaamide paigaldaja, tegevusteks on broneeringu detailide nägemine ning broneeringu paigaldamise staatuse uuendamine (kasutajalood 20-21)

Kasutaja autentimise kasutajalugu on identifikaatoriga 22 ning puudutab kõiki kasutajarolle. Eeposed on välja toodud töö lisade all (Lisa 2).

3.4 Mittefunktsionaalsete nõuete kirjeldus

Mittefunktsionaalseteks nõueteks nimetatakse nõudeid, mis kirjeldavad ootuseid infosüsteemile, kuid ei puuduta äriloogikat. Tavaliselt on need nõudmised, mis seatakse infosüsteemi jõudlusele, skaleeritavusele, turvalisusele, kasutatavusele ning hooldatavusele [20]. Kasutajaliidese arendusel olulised mittefunktsionaalsed nõuded on:

- Sujuv integratsioon tagarakendusega
- Rollipõhine andmetele ligipääs
- Vastuvõtlik hilisematele edasiarendustele
- Dünaamiline ja kohanduv disain
- Interaktiivne andmete visualiseerimine

3.5 Tehniline analüüs

Tehnilise analüüsi alapeatükis põhjendab autor valikuid, mida ta eesmärgipärase lahenduse loomiseks kasutada plaanib. Põhjendatakse valikud, mis puudutavad

[1] Toimingute all on autor mõelnud lisamist, detailide vaatamist ja haldamist

kasutajaliidese arhitektuuri, arenduskeelt ja raamistikku, arendust toetavaid teeke, andme- ja olekuhaldust ning arenduse infrastruktuuri.

3.5.1 Arhitektuur

Lõputöö eesmärgile vastav terviklik lahendus luuakse kahe lõputöö raames, millest üks käsitleb infosüsteemi tagarakendust ning käesolev töö kasutajaliidest. Töö autor on tagarakenduse arendajaga kokku leppinud, et tagarakendus ja kasutajaliides ehitatakse üles klient-server (ingl *client-server)* mudeli järgi, kus kasutajaliides vastutab kliendi päringute edastamise eest serveri poolele. Kokkuleppel kasutatakse esi- ja tagarakenduse suhtluseks RESTful API-t, mis seab andmevahetuseks kindla standardi ning parandab andmevahetuse loetavust. RESTful API (*Application Programming Interface*) on rakendusliides, mis põhineb REST (*Representational State Transfer*) arhitektuuristiilil, mida kasutatakse tavaliselt veebirakenduste kavandamisel. REST-i peamised viis põhimõtet on: kliendi ja serveri eraldamine; oleku puudumine; ühtne liides (päringute edastamisel); puhverdamise võimalus; kihiline arhitektuur [21].

Kasutajaliidese arendamiseks kasutatakse komponendi-põhist (CDD) arhitektuuri. See tähendab, et tervete veebilehtede kirjutamise asemel luuakse kindla eesmärgiga ja taaskasutatavaid komponente. Komponendi-põhine lähenemine teeb rakenduse koodi loetavamaks, parandab taaskasutatavust, parandab veebilehe jõudlust ning teeb kasutajaliidese paremini skaleeritavaks [22, pp. 468-469].

3.5.2 Arenduskeel ja raamistik

Antud kolmekihilise rakenduse kasutajaliidese loomisel on valitud arenduskeeleks TypeScript, mis on tüübipõhine edasiarendus JavaScriptist [23]. Arendusraamistiku valimisel hindas töö autor valiku langetamisel peamiselt kolme raamistiku omadust. Esiteks õppimiskurvi, koos sellega ka dokumentatsiooni. Piiratud aja tõttu oli tähtis, et raamistikuga tutvumine ja sellest piisav arusaam ei tekitaks liiga suurt ajalist kuju. Teiseks hindas autor raamistiku populaarsust, et probleemide esinemise korral oleks autoril suurem võimalus leida kiiresti lahendus. Kolmas aspekt oli autori kogemus antud raamistikuga. Sarnaselt esimese punktiga on tingitud see ajalisest piirangust. Autori kogemus raamistikuga vähendab samamoodi ajalist kulu, mis probleemide lahendamisele, projekti ettevalmistusele ja muule sarnasele võib minna. Populaarsuse aspektist tulenevalt on võrdluseks valitud kaks raamistikku – Vue.js ja Angular – ning

üks teek, React. Võrdluse puhul React-ist raamistikuna rääkides peab autor silmas erinevaid React-il põhinevaid raamistikke, nagu näiteks Next.js [24] ning Gatsby [25].

Esimesena on vaatluse all React-il põhinevad raamistikud. React-i puhul tuuakse peamise probleemina välja pikk õppimiskõver, mis on seotud peamiselt React-i keerulise ökosüsteemiga. Kuigi mõlemas mainitud raamistikus on olemas failipõhine *routing*, puudub neil olemasolev globaalne (rakenduse-ülene) oleku haldamine (ingl *state management*). Seega eduka arendusprotsessi jaoks on vaja aru saada nii React-ist kui teegist, selle põhjal arendatud raamistikust ning lisaks veel välistest teekidest, mis on seotud näiteks oleku haldamisega. Dokumentatsioon React-il endal [26] ning ka sellel põhinevatel raamistikel [27], [24] on autori hinnangul arusaadav, kvaliteetne ning üsna ekstensiivne. Töö autoril on küll marginaalne kogemus näiteks Next.js raamistikuga, kuid tema hinnangul ei ole see ajapiirangu tõttu arvestatava suurusega.

Angular on Google poolt arendatud raamistik, mis sarnaselt Vue.js-le keskendub dünaamiliste ühe-lehe rakenduste (ingl *single-page application*) loomisele. Mõlemal raamistikul on olemas võimekused, mis on antud kasutajaliidese loomisel vajalikud. Nende hulka kuulub näiteks *routing*, rakenduse oleku haldamine ja kahe-suunaline andmevahetus [28], [29], mis teeb andmevahetuse haldamise arusaadavamaks. Seega ajapiirangu tõttu muutub otsustavaks kahe sarnase võimekusega raamistiku vahel valides autori kogemus nendega. Autori tööalase kogemuse pärast valib ta arenduseks Vue.js raamistiku. Allpool on toodud välja ka kokkuvõttev tabel, mis visualiseerib raamistike puudujäägid ning eelised. Õppimiskõverat on kirjeldatud ajapiirangu suhtes, ülejäänud kategooriaid täitmise puhul +-ga.

Tabel 1. Arendusraamistike võrdlus.

	React (Next.js, Gatsby)	**Angular**	**Vue.js**
Dokumentatsioon	+	+	+
Routing	Välised teegid	+	+
Globaalne olekuhaldus	Välised teegid	+	+
Kahe-suunaline andmevahetus		+	+
Arendaja kogemus			+

3.5.3 Toetavad teegid

Kasutajaliidese arendusel kasutatud teegid on seotud peamiselt andmevahetuse, olekuhalduse ning *routing*-uga seotud. Levinud teekide kasutamise eesmärk on taaskasutada olemasolevat koodi ning genereerida näiteks andmevahetuseks kasutajaliidese poolsed API kliendimoodulid, et vältida nende manuaalse kirjutamisega kaasnevat ajakulu ning eksimisvõimalust

- OpenAPI Generator. Võimalik kasutada kasutajaliidese arendusel API kliendimoodulite automaatseks genereerimiseks, andes sellele ette tagarakenduse OpenAPI spetsifikatsiooni [30]. Antud projektis kasutati generaatori Typescript-fetch varianti, mis tagas kliendimoodulite tüübipõhise genereerimise, seejuures kasutades päringute saamiseks Fetch APIt [31]
- Pinia. Spetsiaalselt Vue.js raamistiku jaoks arendatud olekuhaldus teek, mida analüüsitakse pikemalt järgmises peatükis.
- Vue-router. Vue.js raamistiku ametlik *routing* teek, mida analüüsitakse samamoodi järgmises peatükis põhjalikumalt.
- Vue-devtools. Brauseri-plugin, mis teeb jooksvalt andmevahetuse jälgimise, olekuhalduse jälgimise ning *routing*-u jälgimise lihtsamaks (viide https://devtools.vuejs.org/getting-started/introduction#what-is-vue-devtools).
- Tailwind CSS. Vue.js raamistikuga integreeritud CSS haldamise teek/raamistik [32]. Tailwind-i alamteekidest kasutatakse ikoonide lisamisel Heroicons teeki.
- Vite ja Vitest. Lokaalseks rakenduse ehitamiseks ja jooksutamiseks kasutatakse Vite-i ehitustööriista, seega oli loogiline valida testimisraamistikuks Vite-i poolt loodud Vitest raamistik [33].
- Mapbox. Reklaamiasukohtade näitamiseks kasutatakse Mapboxi loodud teeki, mis on spetsiaalselt loodud Vue.js raamistikus kirjutatud projektide jaoks. [34]

3.5.4 Olekuhaldus ja *routing*

Andmete korrektseks haldamiseks ning tagarakendusega sünkroniseerimiseks on kasutajaliidese arenduse puhul vajalik olekuhaldusega tegeleda. Efektiivse olekuhalduse jaoks on otsustanud autor kasutada spetsiaalselt Vue.js raamistiku jaoks arendatud Pinia olekuhaldusese teeki. Pinia töötab andmepoodide (ingl *stores*) põhimõttel, kus üks pood

vastutab tavaliselt ühe kindla andmeedastusobjektiga seotud toimingute eest. Sellisel viisil toimub andmeedastusobjektidega toimingute tegemine ühtse raamistiku järgi, mis muudab andmevahetusprotsessi arendaja jaoks paremini arusaadavamaks. Pinia on ka eos juba tüübitud, mis tähendab, et poodide loomisel tulevad andmetüübi vead jooksvalt kohe välja. Üldiselt on tegemist kerge ning efektiivselt integreeritud olekuhaldusteegiga, mis teeb Vue.js raamistikus olekuhalduse arendaja jaoks palju arusaadavamaks. [35]

Kasutajaliidese arendus toimub komponendipõhise ja SPA (*Single-page application*) arenduspõhimõtete järgi, seega on vajalik *routing*-u kasutamine. Selleks kasutatakse antud projekti puhul Vue.js raamistiku ametlikku teeki, Vue Router. Vue Router toetab dünaamilist *routing*-ut, mis on oluline URL-ide parameetrite käsitlemisel, kapseldunud (ingl *nested*) aadresse, mis võimaldab luua keerulisi kasutajaliidese hierarhiaid. Samuti sisaldab see täiustatud funktsioone, nagu laisk laadimine, mis parandab rakenduse jõudlust, laadides ainult praeguse aadressi jaoks vajalikke komponente, ja navigatsioonikaitsjad, mis võimaldavad kontrollida juurdepääsu aadressitele vastavalt tingimustele, näiteks autentimise staatusele. Need funktsioonid tagavad kasutajatele sujuva ja turvalise navigeerimiskogemuse. [36]

3.5.5 Arenduse infrastruktuur

Tööprotsessi efektiivsemaks kulgemiseks on autoril vaja valida enne arenduse algust kolm tööriista: IDE (*Integrated Development Environment*) arenduseks, versioonihalduse süsteem ning lokaalseks käivitamiseks keskkond konteinerdamiseks. IDE-ks valiti töö kirjutamiseks WebStorm, versioonihalduseks GitHub ning rakenduse konteinerdamiseks Docker. Järgnevalt põhjendatakse iga valikut lühidalt.

WebStorm on arenduskeskkond, mis on mõeldud JavaScriptile põhinevate projektide arendamiseks. Sarnaselt teiste JetBrains'i poolt loodud IDE-dega (millest autoril on kogemusi näiteks Rider-i, IntelliJ IDEA ning PyCharm-iga) on WebStrorm-il intelligentse kooditäiendamise võimekus, jooksev vigade tuvastamine, integreeritud versioonihaldus ning muud sisseehitatud tööriistad, mis tõstavad arendaja efektiivsust. WebStorm toetab ka levinumaid JavaScriptil põhinevaid raamistikke, sh Vue.js, mis teeb projekti esialgse genereerimise lihtsamaks [37]. Välja toodud funktsionaalsuse ning sarnaste IDE-de kasutamiskogemuse tõttu on autor otsustanud arendamiseks WebStorm-i valida.

Git-i põhjal versioonihaldust pakkuvad platvormid on enamjaolt sama funktsionaalsusega, erinedes natukene oma kasutajakogemuse ning kasutajaliidese poolt. Eelneva kogemuse pärast otsustas autor kasutada versioonihalduseks GitHub-i. GitHubi lai kasutajaskond teeb ka esinevate probleemide lahendamist kiiremaks.

Kohalikuks rakenduse käitamiseks on kasutusel Docker. Docker on platvorm, mis võimaldab rakenduste kohaliku käitamise kergete, isoleeritud konteinerite sees. Need konteinerid pakendavad rakenduse koos selle sõltuvustega, tagades järjepideva käitumise erinevates keskkondades, alates arendusest kuni tootmiseni. Eraldades rakendused konteineritesse, suurendab Docker skaleeritavust, lihtsustab juurutamisprotsesse ja tõhustab arendustööde voogusid. [38]

4 Kasutajaliidese arendus

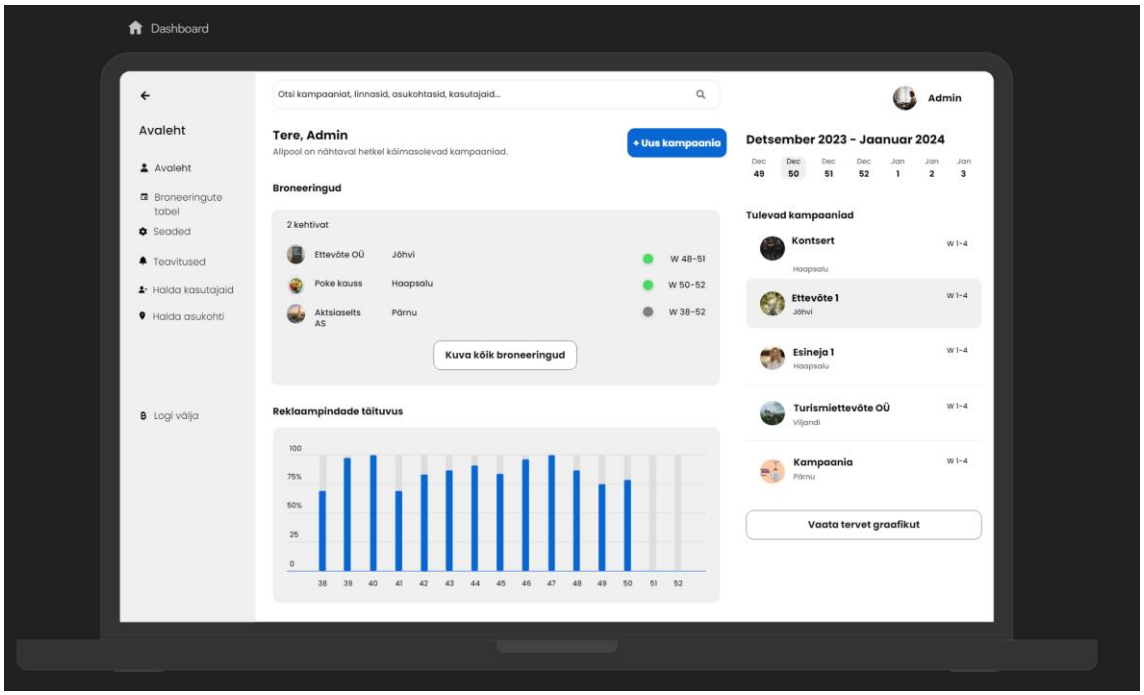
Lõputöö praktilises osas kirjeldatakse peamiselt arendusprotsessi erinevaid etappe. Lähemalt vaadatakse arenduse ettevalmistust, kasutajaliidese enda arendust, dokumentatsiooni koostamist ning kasutajaliidese testimisega seonduvat. Praktilises osas tugineb autor analüütilises peatükis välja toodud põhimõtetele ja nõuetele ning kasutab arendamiseks tehnoloogiaid, mida analüüsis ning põhjendas tehnilise analüüsi peatükis.

4.1 Arenduse etapid ja osad

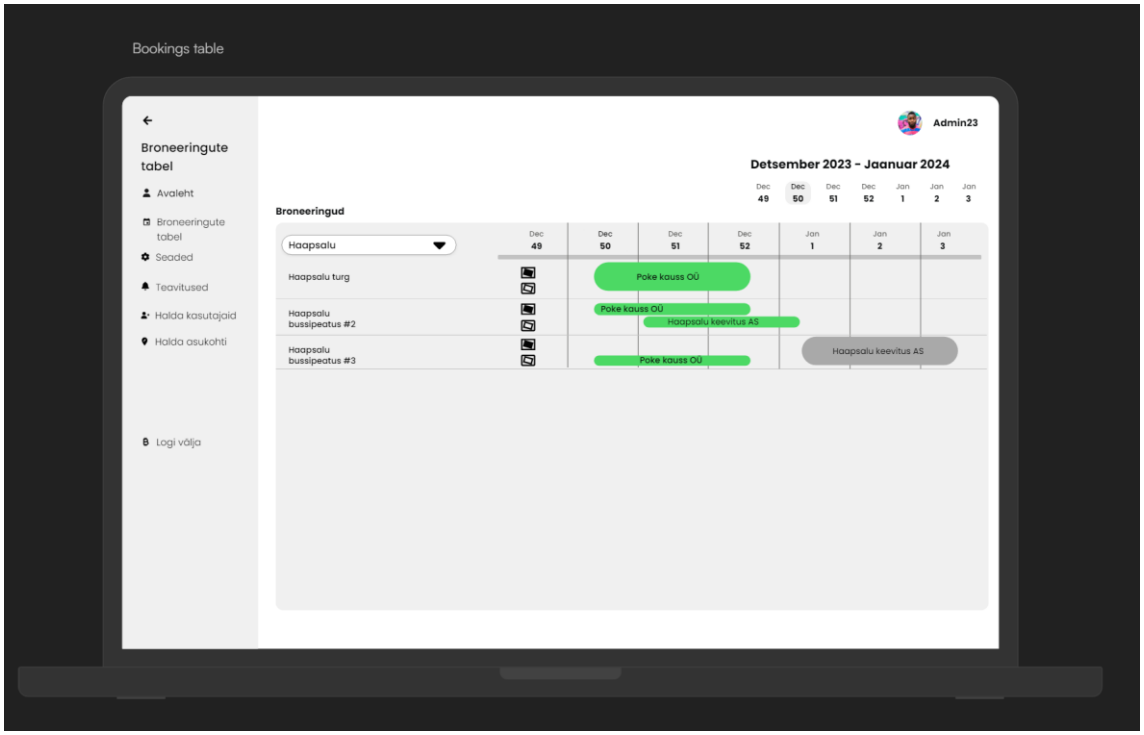
Järgmistes alapeatükkides röögib autor detailsemalt erinevatest töö etappidest ning projekti osadest, mille eesmärk on luua kasutajaliidese jaoks toimiv infrastruktuur ning lõpuks toimiv kasutajaliides. Lähemalt tutvustatakse nelja erinevat arenduse osa, alustades ettevalmistusest, mille järel tulevad peamised arendustööde grupid. Nendeks on andmepäringute (API kliendimoodulite koostamine), andme- ja olekuhaldus (Pinia poodide loomine) ning üldine komponentide loomise loogika, nii rollipõhiste kui ka taaskasutatavate komponentide kohta.

4.1.1 Prototüübitud vaated

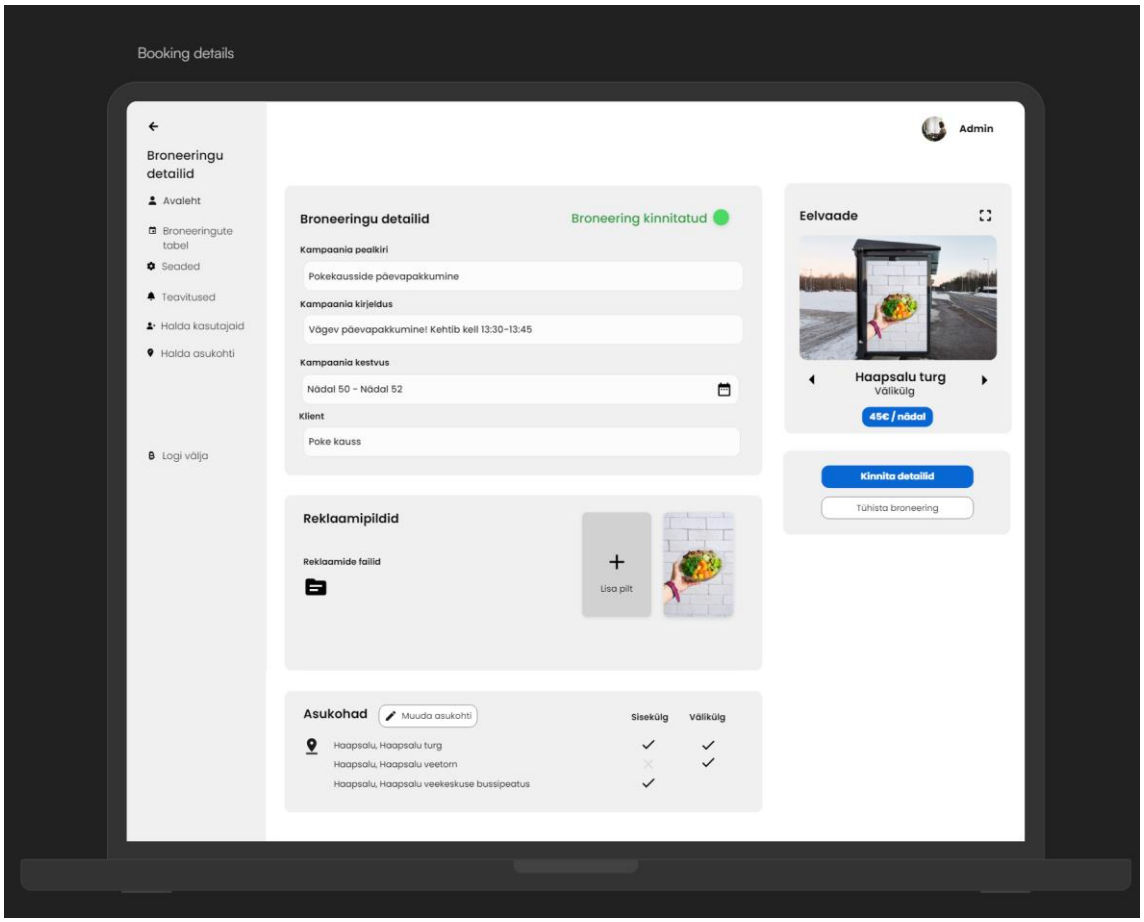
Kasutajaliidese arenduse toetamiseks prototüübiti rakenduse peamised vaated, et paika panna kasutajaliidese üldine stiil ning kooskõlastada tähtsamad elementide paigutused ja rakenduse põhivoog ettevõttega. Prototüübi loomiseks kasutati Uizard-it, mis on tehisintellektiga varustatud tarkvara prototüüpide loomiseks [39]. Tehisintellekti abiga loodi vaadetele põhjad, millele kujundati ja lisati vastavalt ettevõtte nõuetele ja kasutajalugudele vajalikud komponendid. Prototüübiti kokku neli vaadet. Esiteks avaleht, mis disainiti töölaua põhimõttel, et anda kasutajale informatiivne ülevaade tähtsamatest broneeringutega seotud andmetest. Järgmisena koostati detailsed broneeringute tabelist, broneeringu detailvaatest ning kasutaja detailvaatest. Prototüübi disainimisel lähtuti minimalistlikest disainipõhimõtetest, et vältida info üleküllust ja olukordi, kus kasutaja ei mõistaks mõne elemendi eesmärki. Järgneval neljal joonisel on näha loodud vaated.



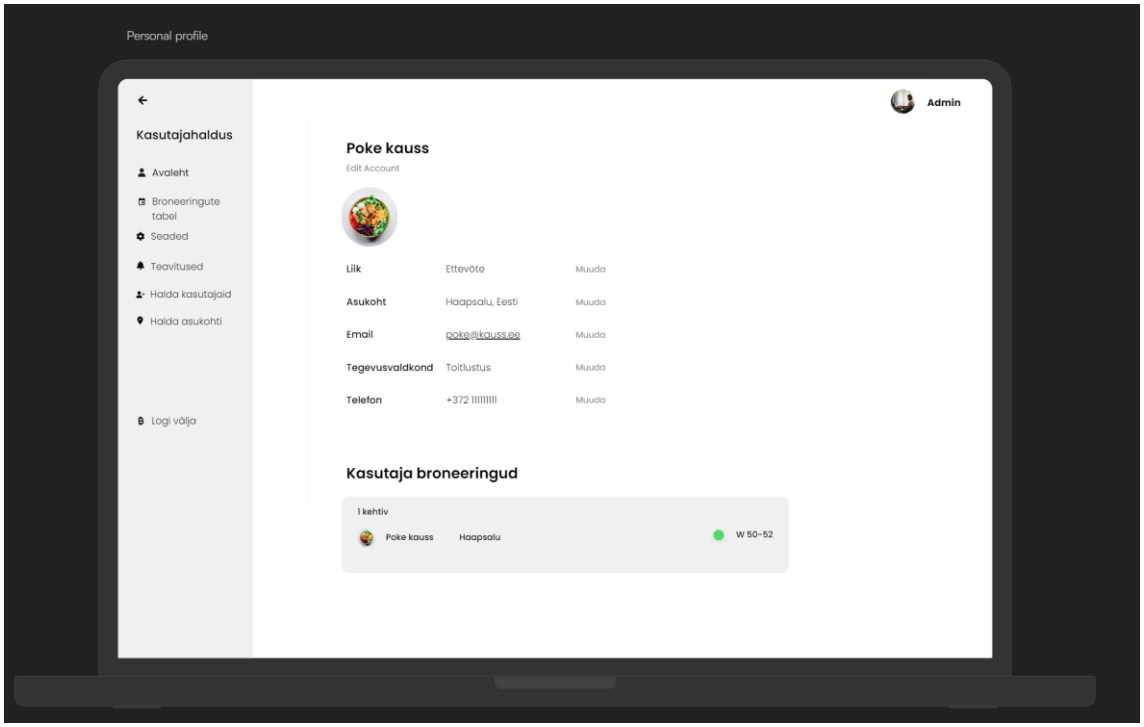
Joonis 1 Prototüübi avalehe vaade.



Joonis 2 Prototüübi broneeringukalendri vaade.



Joonis 3 Prototüübi broneeringu detailandmete vaade.



Joonis 4 Protüübi kliendi detailandmete vaade.

4.1.2 Ettevalmistus arenduseks

Esimesena luuakse WebStorm-i pakutavaid valikuid kasutades Vue-rakenduse põhi, millele lisatakse esialgu vajaolevad sõltuvused rakenduse lokaalseks jooksutamiseks. Seejärel seotakse projekt GitHubi koodihoidlaga, et arenduse käigus edukalt versioonihaldust teostada. Seejärel tühjendatakse projekt esialgu mittevajalikest failidest ning kaustadest. Enne arendustöö algust lisatakse esimesed vajalikud teegid ja sõltuvused, mis on loetletud peatükis 3.5.3. Teekide eesmärgid on samuti seal välja toodud. Pärast nimetatud teekide lisamist on projekt arendusöödeks valmis.

4.1.3 Andmepäringud

Korrektsete andmepäringute koostamiseks kasutati ära OpenAPI Generator-i TypeScript-Fetch varianti, mis genereeris vastavalt tagarakenduse OpenAPI spetsifikatsioonile kasutajaliidesele API kliendimooduli. Generaatori kasutamise eesmärgiks oli aja kokkuhoid ning ebakõlade vältimine, mis moodulite käsitsi kirjutamisel oleks töö ajalise piirangu tõttu tõsiseks probleemiks osutunud. Generaatori TypeScript variandi kasutamine kindlustas ka selle, et päringud on korrektselt tüübitud. See tähendab ka seda, et otspunktide poole pöörduvat koodi kirjutades on arendajal väiksem võimalus kasutada valesid andmetüüpe. Õige seadistamise ja automatiseerimise korral on võimalik vastavalt tagarakenduse uuendustele sünkroonis hoida kasutajaliidese kliendimoodul [40]. Läbi selle on võimalik suuresti parandada koodi hallatavust ja hooldatavust, mis eriti tüübikoodi (ingl *boilerplate code*) loomisel on väga tähtis. Joonisel 5 on toodud näide kasutajaliidese kliendimooduli toorest meetodist, mis on otseseks ühenduskohaks kasutajaliidese ja tagarakenduse vahel. Ülejäänud kliendimoodulite toorete meetodite ülesehitus on samasugune. Alustatakse sisestatud andmete valideerimisega ning andmete JSON-kujule viimisega. Sisendandmete valideerimisele järgneb JWT pääsutalongi ning muude päringupäiste lisamine päringule. Meetod lõppeb päringu saatmise ning vastuse ootamisega, pärast mida tagastatakse päringu tulemus. Toore meetodi ning Pinia poodides kasutatava ’esindus’-meetodi eesmärk on teha lihtsamaks spetsiifilise andmepäringuga mitte-seotud andmete jälgimine (nö päringu üldandmed).

```
async getReservationsToInstallRaw(requestParameters:
GetReservationsToInstallRequest, initOverrides?: RequestInit |
runtime.InitOverrideFunction):
Promise<runtime.ApiResponse<Array<PublicReservation>>> {
    const queryParameters: any = {};

    if (requestParameters['searchAreaCenterPointLatitude'] != null) {
        queryParameters['Search area center point latitude'] =
requestParameters['searchAreaCenterPointLatitude'];
    }

    ...

    const headerParameters: runtime.HTTPHeaders = {};

    if (this.configuration && this.configuration.accessToken) {
        const token = this.configuration.accessToken;
        const tokenString = await token("Bearer Authentication", []);

        if (tokenString) {
            headerParameters["Authorization"] = `Bearer
${tokenString}`;
        }
    }
    const response = await this.request({
        path: `/api/install`,
        method: 'GET',
        headers: headerParameters,
        query: queryParameters,
    }, initOverrides);

    return new runtime.JSONApiResponse(response, (jsonValue) =>
jsonValue.map(PublicReservationFromJSON));
}
```

Joonis 5. Koodinäide API kliendimooduli toorest meetodist.

Toore meetodi kutsub välja esindusmeetod, mille näide on toodud Joonisel 2. Esindusmeetodi ja toore meetodi põhjendus on juba eelnevalt välja toodud.

```
 async getReservationsToInstall(requestParameters:
GetReservationsToInstallRequest = {}, initOverrides?: RequestInit |
runtime.InitOverrideFunction): Promise<Array<PublicReservation>> {
    const response = await
this.getReservationsToInstallRaw(requestParameters, initOverrides);
    return await response.value();
}
```

Joonis 6. Koodinäide API kliendimooduli esindusmeetodist.

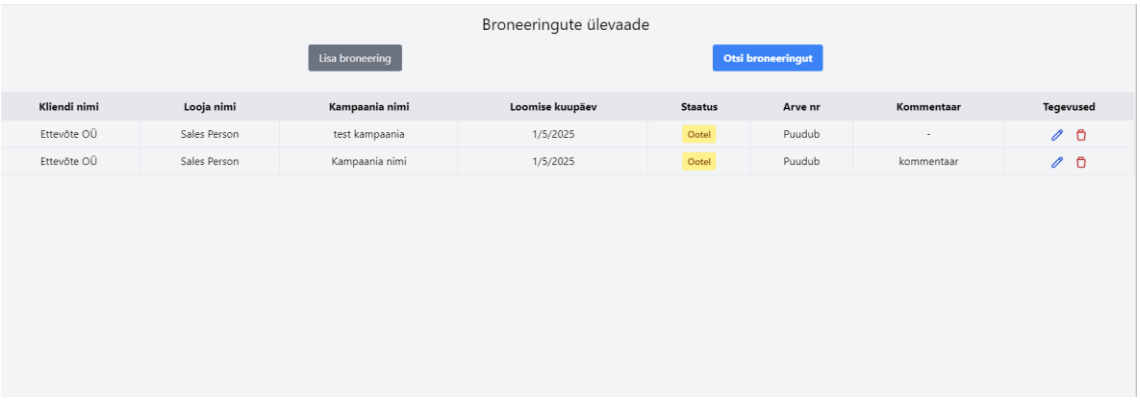
4.1.4 Olekuhaldus

Efektiivne olekuhaldus on tähtis osa kasutajaliidese infrastruktuurist. Selle tõttu valis autor arenduseks laialt kasutusel oleva, detailse dokumentatsiooni ning suure kogukonnaga olekuhaldusteegi Pinia. Olekuhaldusega tagatakse, et andmestik, mis on kasutusel rakenduse erinevates osades, oleks omavahel sünkroniseeritud. Pinia andmepoe koostamisel on autor pidanud oluliseks andmepoe koodi arusaadavust ning hallatavust. Andmepoe meetodid ning andmetüüpide grupeeringud on koostatud silmas pidades seda, et omavahel seotud andmete töötlemine oleks jätkusuutlik ning arusaadav. Meetodeid lihtsana hoides säilib ka rakendusel parem silutavus, mis on arendusjärgus oleva projekti jaoks oluline. Arenduse käigus osutus fundamentaalse tähtsusega Pinia andmepoeks autentimisega seotud andmepood, mis lihtsustas kasutajaliideses navigeerimist ning rollipõhiste andmepäringute saatmist. Lisaks sellele aitas mitme väiksema ja spetsiiflise Pinia andmepoe loomine ühe suurema API kliendimooduli (näiteks broneeringutega seotud API kliendimooduli) loogilisteks osadeks jagada, mis tegi kasutajaliidese arendusel andmete haldamise ja nende välja kuvamise palju lihtsamaks. Läbi ühe kliendimooduli mitmeks Pinia poeks jagamise õnnestus vältida ka olukorda, kus ühes kohas läheks erinevate andmete hulk nii arenduse kui ka kasutaja vaatest mitte-hallatavaks.

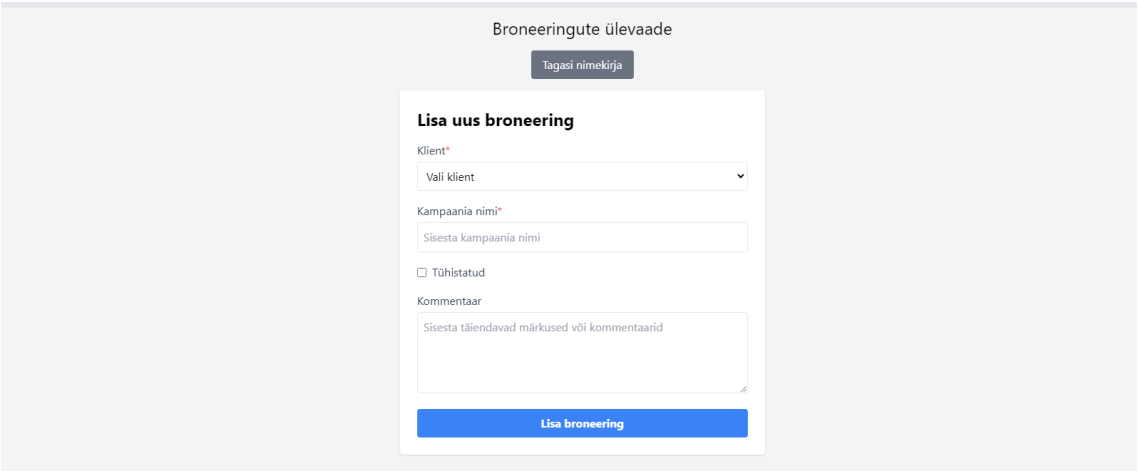
4.1.5 Rollipõhised ja üldised komponendid

Rakenduse komponentide loomisel on peamiselt lähtutud kahest põhimõttest. Esiteks on koostatud üldised komponendid, mis asuvad enamjaolt komponentide puus juure pool. Rollispetsiifilised komponendid pakuvad üldjuhul mingit kindlat funktsionaalsust või ligipääsu andmetele, millele aktiivsel kasutajarollil õigus on. Nagu varasemalt mainitud, on administraatoreid puudutavad komponendid seotud kasutajate haldusega,

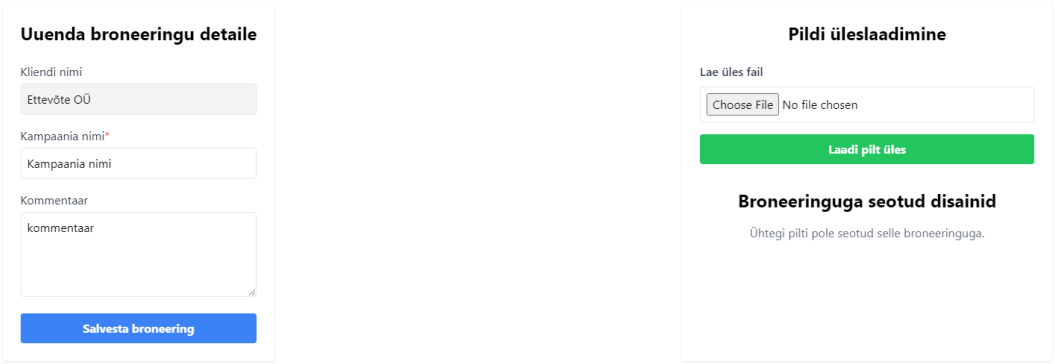
müügipersonali puudutavad komponendid serveerivad broneeringu- ja reklaamihaldusega soetud funktsionaalsust ning paigaldajatega seotud komponendid broneeringu staatusega soetud tegevusi. Üldiseid komponente kasutatakse kahel eesmärgil. Esiteks alusena rollipõhistele komponentidele, teiseks komponentidena, mis aitavad kuvada informatsiooni, kuid on korduvkasutatavad läbi erinevate rollide. Komponente disainides on järgitud funktsionaalseid ning mittefunktsionaalseid nõudeid. Allpool on toodud näidetena broneeringute haldamisega seotud vaated, millele sarnaselt on üles ehitatud ka muude andmeobjektide vaated.



Joonis 7. Broneeringute ülevaade.



Joonis 8. Esialgne broneeringu lisamise vorm.



Joonis 9. Broneeringu detailide muutmine koos piltide lisamise funktsionaalsusega.

4.2 Dokumentatsioon

Projekti puhul dokumenteeriti peamiselt kahte aspekti. Esiteks loodi VitePressi generaatoriga [41] kasutajaliidesele üldine dokumentatsioon, mis tutvustab rakendust ning aitab näiteks selle lokaalse paigaldamisega. Komponentide dokumenteerimiseks plaanitakse kasutada Storybook raamistikku, mis on mõeldud komponentide isoleeritud arenduseks, testimiseks ning dokumenteerimiseks [42]. Lisaks sellele on API kliendimoodulile genereeritud kokkuvõttev dokumentatsioon OpenAI generaatori poolt.

4.3 Testimine

Kasutajaliidese tähtsamatele funktsionaalsetele komponentidele on kirjutatud testid kasutades Vitest raamistikku. Peamiselt jagunevad komponendi testid kahte kategooriasse. Esiteks testitakse komponentide *render*-dumist ja rollipõhiselt kuvatud alamkomponente. Esimest tüüpi testid vastutavad selle eest, et komponentide visuaalne pool (vähemalt teoorias) oleks korrektne. Teisena testitakse komponentide funktsionaalsust. Näiteks andmete kuvamiseks mõeldud komponentide puhul antakse neile kindel andmestik simuleeritud API otspunktide poolt ning kontrollitakse, kas selle põhjal loodud vaade on vähemalt HTML-i poolest õige.

4.4 Paigaldamine

Arendustööde käigus kasutatakse lokaalset paigaldust. Selleks on vajalik esiteks Dockeri ning mõne Java programmeerimiskeelt toetava IDE olemasolu, et jooksutada lokaalset

instantsi tagarakendusest. Arendustööde kiirendamiseks muudeti ajutiselt tagarakenduse CORS poliitikat, et tagada kasutajaliidesest saadetud päringute õnnestumine. Kasutajaliidese kohalikuks käivitamiseks on vajalik Vue.js raamistikku toetav IDE ning npm (Node Package Manager), jooksutades käsku *npm run dev*. Broneeringute haldussüsteemi püsiv ülespanek lepitakse kokku tagarakenduse loojaga pärast kasutajaliidese kasutusvalmis seisundi saavutamist.

5 Tulemused

Tulemuste peatükk koosneb kolmest erinevast osast. Esimesena annab autor omapoolse hinnangu sellele, kuidas ta on lõputöö raames saavutatud tulemusega rahul. See tähendab nii analüütilise kui ka praktilise osa hindamist. Autori hinnangule järgneb Kasutajate tagasiside esimese arendustsükli kohta. Kasutajate tagasiside kogutakse demo põhjal, mille jooksul demonstreeritakse loodud rakenduse funktsionaalsust erinevates rollides ning üldist väljanägemist. Hinnangute põhjal pakutakse välja võimalused edasiarendusteks, esmalt keskendudes arendustele ja parandustele, mida oleks vaja kohe teostada ning seejärel tuuakse välja funktsionaalsus, mida võiks lähitulevikus hakata järk-järgult lisama.

5.1 Autoripoolne hinnang

Autor annab omapoolse hinnangu vastavalt peatükis 2.5 seatud eesmärkidele, mida antud lõputööga saavutada sooviti. Hinnang antakse jaotatakse kaheks, analüütilise osa ja praktilise osa hinnanguks. Analüütilise osa hinnangus avaldab autor oma arvamust kogutud informatsiooni kohta ning hindab, kuidas aitas see formuleerida töö lõpptulemuse. Praktilise osa hindamisel võtab autor aluseks funktsionaalsed ja mitte-funktsionaalsed nõuded ning annab neile hinnangu koos põhjendusega.

5.1.1 Autori hinnang analüütilisele osale

Töö eesmärkides nimetati lõputöö analüütilise osa kolm peamist eesmärki. Esimesena toodi välja olemasolevate lahenduste analüüs. Autori hinnangul saavutati seatud eesmärk, sest näitena toodi lahendusi kolmest erinevast kategooriast ning analüüsi käigus selgitati välja nende eelised ja kitsaskohad. Antud protsess aitas luua parema ettekujutuse, millised on peamised kitsaskohad olemasolevatel süsteemidel ning kuidas langetada ratsionaalne otsus selliste olemasolevate lahenduste suhtes, mis idee poolest täidab funktsionaalseid ja mitte-funktsionaalseid nõudeid.

Teise analüütilise osa eesmärgina mainiti põhjalikku ärianalüüsi, mille alla käis ka üldiste disainipõhimõtete analüüs. Autori hinnangul aitasid saavutada selle eesmärgi peatükkides

3.2 kuni 3.4 tehtud analüüs. Disainipõhimõtete analüüsis toodi välja põhimõtted, mis küll esmapilgul võivad tunduda iseenesest mõistetavad, kuid kui neile tähelepanu ei pöörata, võib rakenduse kasutatavus tõsiselt kannatada. Funktsionaalsete ja mittefunktsionaalsete nõuete analüüs oli samuti edukas, sest pani paika konkreetse raami, millest arendustöid tehes lähtuda. Põhjalikum analüüs rakenduse nõuetele vastavuse kohta kirjutatakse lahti järgmises alapeatükis.

Kolmanda ja viimasena toodi välja vajadus põhjaliku tehnilise analüüsi järele. Autori hinnangul osutus see osa samuti edukaks, sest peatükis 3.5 tehtud analüüsi käigus otsustati ära projekti tehnilised aspektid, mis loodetavasti viivad positiivse praktilise tulemuseni.

5.1.2 Autori hinnang praktilisele osale

Autoripoolne hinnang lõputöö praktilisele osale on kahetine. Ühest küljest saavutati rahuldav tulemus funktsionaalsete nõuete täitmisega, mis viitab piisavale kasutajaliidese infrastruktuuri loomisele, kuid tulemuste kirjeldamise ajaks ei täidetud autori hinnangul kõiki mitte-funktsionaalseid nõudeid, mis lõputöö ülesandepüstitust kirjeldades seati. Funktsionaalsetest nõuetest kaeti ära põhiline ning lahendati ka failide säilitamise ja interaktiivse kaardi probleemid, mida töö alguses peamiste kitsaskohtadena kirjeldati. Kuid seejuures ei ole autor rahul sellega, et mitte-funktsionaalsetest nõuetest jäid lõputöö kirjutamise ajal saavutamata osaliselt sujuv integratsioon tagarakendusega (püsiva paigalduse probleem) ning esteetilise välimusega andmete visualiseerimine.

Autor jääb aga rahule tööprotsessi endaga, sest kasutajaliidese arendamisel loodud projekti ehitamine aitas autoril saada parema arusaama sellest, milline võiks olla ühe kasutajaliidese infrastruktuur. Kasutajaliidest edasi arendades soovib autor pöörata suuremat tähelepanu elementide disainile.

5.2 Kasutajate esmane tagasiside

Tagasiside koguti ettevõtte kasutajalt ja kolmandalt isikult (kellel on taust IT-süsteemide arenduses), et saada nii tavakasutaja kui ka asjatundja. Tagasiside koguti demo ning piiratud testimise pealt. Ettevõtte kasutajalt tulnud tagasiside põhjal veendus autor, et tööga on saavutatud funktsionaalsed nõuded ning selle tarbeks tehtud analüüside põhjal on suudetud hoida kasutajaliides olemasolevale tööprotsessile piisavalt sarnane. Kiideti

ka omavahel seotud andmete koos kuvamist, näiteks broneeringu ja sellega seotud disaini. Kolmandalt isikult saadud tagasiside oli suunatud rohkem disainile ning elementide paigutusele. Äriloogika ning kasutusvoo kohta etteheiteid ei olnud, sellele eelnes ka rakenduse äriloogika tutvustamine.

5.3 Edasiarenduse võimalused

Esimesena võtab autor arvesse kasutajatelt tulnud tagasisidet, millest enamuse moodustavad kolmandalt isikult kogutud kommentaarid. Seega edasiarenduse esimeseks prioriteediks on kasutajaliidese väljanägemise parandamine. Pärast disainiparanduste tegemist on järgmiseks prioriteediks infosüsteemi püsiv paigaldamine koostöös tagarakenduse arendajaga. Pärast infosüsteemi püsivat paigaldamist ja kasutajatele üle andmist on võimalik arendajatel jätkata infosüsteemi täiustamisega lähtuvalt ideedest, mis tulid lauale ärianalüüsi käigus. Üks neist on näiteks infosüsteemi integreerimine raamatupidamistarkvaraga Scoro, mis on hetkel ettevõttes arvete haldamiseks kasutusel. Tänu tagarakenduse abstraktsusele on võimalik laiendada platvormi ka digitaalreklaami pakkujatele.

6 Kokkuvõte

Töö eesmärk oli välireklaamindusega soetud ettevõttele loodava infosüsteemi kasutajaliidese analüüs ning arendus, mis hiljem paigaldatakse püsivalt koos tagarakendusega.

Töö käigus kirjeldati ära probleem millega tegeleti, lahendus sellele probleemile ning mis on lahenduse eesmärgid. Kirjeldati ära ka töö üldised eesmärgid. Ülesandepüstituse ning Probleemi analüüs peatükkides kirjeldati edukalt probleemi olemus ning koostati sellele kavandatava lahenduse analüüs, mis autori hinnangul saavutas kõik seatud eesmärgid.

Kasutajaliidese arenduse puhul jõuti küll täita seatud funktsionaalsed nõuded, kuid autori kui ka tagasiside põhjal saab järeldada, et ei saavutatud sajaprotsendiliselt eesmärke, mis ülesandepüstituses töö praktilisele osale seati. Järgmised prioriteedid kasutajaliidese arenduseks on seatud, et täita töö esialgne praktiline eesmärk ning anda üle ettevõttele töötav ning eeskujulik infosüsteem.

Üldiselt vastavad töö tulemused suuresti sellele, mida autor tööd koostama hakates ette nägi. Autor jääb rahule töö analüütilise osa eesmärkide saavutamisega, kuid näeb arenguruumi arendustööd puudutavates aspektides, eriti kasutajaliidese väljanägemise poolest. Edasiarenduse plaanide kohaselt peaks lähiajal valmis saama lõplik versioon kasutajaliidesest, mida on võimalik koos tagarakendusega valdkonnaspetsiifilise tarkvarana jagada.

Kasutatud kirjandus

[1] J. U. Sütt, „Välireklaami haldussüsteemi tagarakenduse arendamine,“ Tallinna Tehnikaülikool, Tallinn, 2024.

[2] Asana, Inc., „Product: Asana,“ [Võrgumaterjal]. Available: https://asana.com/product. [Kasutatud 07 11 2024].

[3] Asana, Inc., „Integrations: Asana,“ Asana, Inc., [Võrgumaterjal]. Available: https://asana.com/integrations. [Kasutatud 07 11 2024].

[4] DOmedia, LLC., „Introduction: DoMedia,“ DOmedia, LLC., [Võrgumaterjal]. Available: https://www.domedia.com/. [Kasutatud 07 11 2024].

[5] LDSK., „Introduction: SignKick,“ LDSK., [Võrgumaterjal]. Available: https://www.signkick.com/. [Kasutatud 12 11 2024].

[6] VIOOH, „Introduction: VIOOH,“ VIOOH, [Võrgumaterjal]. Available: https://www.viooh.com/. [Kasutatud 12 11 2024].

[7] BillboardPlanet LLC., „Introduction: Quantum software,“ BillboardPlanet LLC., [Võrgumaterjal]. Available: https://online.billboardplanet.com/quantum/. [Kasutatud 21 10 2024].

[8] World Wide Web Consortium, „Web Content Accessibility Guidelines (WCAG) 2.2,“ 05 10 2023. [Võrgumaterjal]. Available: https://www.w3.org/TR/WCAG22/. [Kasutatud 24 11 2024].

[9] Siteimprove A/S, „Website Accessibility Checker,“ Siteimprove A/S, [Võrgumaterjal]. Available: https://www.siteimprove.com/toolkit/accessibility-checker/. [Kasutatud 24 11 2024].

[10] Google Inc., „Understanding Core Web Vitals and Google search results,“ Google Inc., 31 10 2024. [Võrgumaterjal]. Available: https://developers.google.com/search/docs/appearance/core-web-vitals. [Kasutatud 24 11 2024].

[11] E. Falode, „Simplicity in Design: Why Less Is Often More,“ 21 10 2024. [Võrgumaterjal]. Available: https://cyberogism.com/simplicity-in-design-philosophy/. [Kasutatud 24 11 2024].

[12] M. Soegaard, „The Role of Micro-interactions in Modern UX,“ 21 03 2024. [Võrgumaterjal]. Available: https://www.interaction-design.org/literature/article/micro-interactions-ux. [Kasutatud 24 11 2024].

[13] A. Hollingsworth, „User Experience Optimization: Guide to Exceptional Online Interactions,“ 13 11 2023. [Võrgumaterjal]. Available: https://www.oyova.com/blog/user-experience-optimization/. [Kasutatud 24 11 2024].

[14] E. Swain, „Does Dark Mode Reduce Eye Strain?,“ 21 02 2024. [Võrgumaterjal]. Available: https://www.visioncenter.org/blog/dark-mode-eye-health/. [Kasutatud 24 11 2024].

[15] A. Sprabary, „Is dark mode better or worse for your eyes?,“ 06 06 2023. [Võrgumaterjal]. Available: https://www.allaboutvision.com/digital-eye-strain/is-dark-mode-better-for-eyes/. [Kasutatud 24 11 2024].

[16] Interaction Design Foundation, „Visual Hierarchy,“ Interaction Design Foundation, [Võrgumaterjal]. Available: https://www.interaction-design.org/literature/topics/visual-hierarchy. [Kasutatud 24 11 2024].

[17] C. Vinney, „The role of gamification in UX design: creating engaging user experiences,“ 09 08 2023. [Võrgumaterjal]. Available: https://www.uxdesigninstitute.com/blog/gamification-in-ux-design/. [Kasutatud 24 11 2024].

[18] Nuclino, „A Guide to Functional Requirements (with Examples),“ Nuclino, [Võrgumaterjal]. Available: https://www.nuclino.com/articles/functional-requirements. [Kasutatud 22 11 2024].

[19] M. Rehkopf, „User stories with examples and a template,“ Atlassian, [Võrgumaterjal]. Available: https://www.atlassian.com/agile/project-management/user-stories. [Kasutatud 24 11 2024].

[20] Scaled Agile, Inc., „Nonfunctional Requirements,“ Scaled Agile, Inc., [Võrgumaterjal]. Available: https://scaledagileframework.com/nonfunctional-requirements/. [Kasutatud 22 11 2024].

[21] L. Gupta, „What is REST?,“ 12 12 2023. [Võrgumaterjal]. Available: https://restfulapi.net/. [Kasutatud 19 11 2024].

[22] P. Gulia, „Component Based Software Development Life Cycle Models: A Comparative Review,“ 2017. [Võrgumaterjal]. Available: https://www.academia.edu/96566585/Component_Based_Software_Development_Life_Cycle_Models_A_Comparative_Review. [Kasutatud 19 11 2024].

[23] Microsoft, „What is TypeScript?,“ Microsoft, [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 18 11 2024].

[24] Vercel, Inc., „Introduction: Next.js,“ Vercel, Inc., [Võrgumaterjal]. Available: https://nextjs.org/docs. [Kasutatud 18 11 2024].

[25] Gatsby, Inc., „Introduction: Gatsby.js,“ Gatsby, Inc., [Võrgumaterjal]. Available: https://www.gatsbyjs.com/front-end-framework/. [Kasutatud 18 11 2024].

[26] Meta Platforms, Inc., „React Reference Overview,“ Meta Platforms, Inc., [Võrgumaterjal]. Available: https://react.dev/reference/react. [Kasutatud 18 11 2024].

[27] Gatsby, Inc., „Documentation: Gatsby.js,“ Gatsby, Inc., [Võrgumaterjal]. Available: https://www.gatsbyjs.com/docs. [Kasutatud 18 11 2024].

[28] Google, Inc. , „What is Angular?,“ Google, Inc. , [Võrgumaterjal]. Available: https://angular.dev/overview. [Kasutatud 18 11 2024].

[29] E. You, „Introduction: Vue.js,“ [Võrgumaterjal]. Available: https://vuejs.org/guide/introduction.html. [Kasutatud 18 11 2024].

[30] OpenAPI-Generator Contributors, „Documentation for the typescript-fetch Generator,“ [Võrgumaterjal]. Available: https://openapi-generator.tech/docs/generators/typescript-fetch/. [Kasutatud 26 11 2024].

[31] Mdn Web Docs, „Using the Fetch API,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch. [Kasutatud 26 11 2024].

[32] Tailwind Labs, Inc., „Get started with Tailwind CSS,“ [Võrgumaterjal]. Available: https://tailwindcss.com/docs/installation. [Kasutatud 27 11 2024].

[33] Anthony Fu, Matías Capeletto and Vitest contributors, „Introduction: Vitest,“ [Võrgumaterjal]. Available: https://vitest.dev/guide. [Kasutatud 27 11 2024].

[34] Mapbox, „Mapbox in a Vue app,“ MapBox, 2024. [Võrgumaterjal]. Available: https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-vue/#set-up-the-vue-app-structure. [Kasutatud 16 12 2024].

[35] E. S. M. Morote, „Pinia: The intuitive store for Vue.js,“ [Võrgumaterjal]. Available: https://pinia.vuejs.org/. [Kasutatud 20 11 2024].

[36] Evan You, Eduardo San Martin Morote, „Vue Router: The official Router for Vue.js,“ [Võrgumaterjal]. Available: https://router.vuejs.org/. [Kasutatud 20 11 2024].

[37] JetBrains s.r.o., „Introduction: WebStorm,“ JetBrains s.r.o., [Võrgumaterjal]. Available: https://www.jetbrains.com/webstorm/. [Kasutatud 20 11 2024].

[38] Docker, Inc., „What is Docker?,“ Docker, Inc., [Võrgumaterjal]. Available: https://docs.docker.com/get-started/docker-overview/. [Kasutatud 20 11 2024].

[39] Uizard Technologies, „Uizard: product,“ Uizard Technologies, [Võrgumaterjal]. Available: https://uizard.io/product/. [Kasutatud 28 11 2024].

[40] OpenAPI-Generator Contributors, „Workflow Integrations,“ [Võrgumaterjal]. Available: https://openapi-generator.tech/docs/integrations/. [Kasutatud 28 11 2024].

[41] E. You, „What is VitePress?,“ [Võrgumaterjal]. Available: https://vitepress.dev/guide/what-is-vitepress. [Kasutatud 28 11 2024].

[42] Storybook, Inc., „Get started with Storybook,“ [Võrgumaterjal]. Available: https://storybook.js.org/docs. [Kasutatud 28 11 2024].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Karl Stefan Lill

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Välireklaamide haldamissüsteemi kasutajaliidese loomine", mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

06.01.2025

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Eeposed ja kasutajalood

Eepos 1: Kasutajate haldamine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
1	administraator	luua süsteemi kasutaja	võimaldada töötajal vastavalt rollile süsteemi oma õiguste piires kasutada
2	administraator	näha kasutajaid	omada ülevaadet süsteemis registreeritud kasutajatest ja nende õigustest
3	administraator	hallata kasutajaid	kasutajaid vajadusel muuta või kustutada

Eepos 2: broneeringute haldamine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
4	müügiagent	lisada broneeringu	jälgida selle edaspidist staatust
5	müügiagent	näha broneeringuid	omada ülevaadet kõikidest broneeringutest ning olla kursis nende detailidega
6	müügiagent	hallata broneeringuid	broneeringuid vajadusel muuta või kustutada

Eepos 3: Kliendiprofiilide haldamine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
7	müügiagent	lisada kasutajaprofiili	broneeringuid siduda kliendiga
8	müügiagent	näha kasutajaprofiile	omada ülevaadet kasutajaprofiilidest ja nende detailidest
9	müügiagent	hallata kasutajaprofiile	kliendi detaile vajadusel muuta või profiil kustutada

Eepos 4: Reklaamiobjektide haldamine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
10	müügiagent	lisada reklaamiasukoha	kasutada seda reklaampindade koondamiseks
11	müügiagent	näha reklaamiasukoha	omada ülevaadet reklaamiasukohtadest ja nende detailidest
12	müügiagent	hallata reklaamiasukoha	reklaamiasukohti vajadusel muuta või kustutada
13	müügiagent	lisada reklaampinna	reklaampinda välja rentida
14	müügiagent	näha reklaampindu	omada ülevaadet reklaampindadest ja nende detailidest
15	müügiagent	hallata reklaampindu	reklaampindu vajadusel muuta või kustutada
16	müügiagent	lisada reklaamiobjektide gruppe	kasutada seda reklaampindade või reklaamiasukohtade grupeerimiseks
17	müügiagent	näha reklaamiobjektide gruppe	omada ülevaadet reklaamiobjektide gruppidest ja nende detailidest
18	müügiagent	hallata reklaamiobjektide gruppe	reklaamiobjektigruppi vajadusel muuta või kustutada

Eepos 5: Reklaamkujunduste edastamine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
19	müügiagent	edastada reklaamkujundused trükikojale	broneeringuga seotud reklaamkujunduste väljatrükid õigeaegselt kätte

Eepos 6: Broneeringu täitmine			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
20	paigaldaja	näha broneeringu detaile	paigaldada õiged reklaamid õigetesse asukohtadesse
21	paigaldaja	uuendada broneeringu paigaldamise staatust	hoida müügiagenti kursis reklaamide paigaldamise staatusega

Universaalne kasutajalugu			
ID	**Kasutajarollis** ***<roll>***	**soovin** ***<tegevus>***	**et saaksin** ***<eesmärk>***
22	administraator, müügiagent, paigaldaja	süsteemi sisse logida	vastavalt kasutajarollile ja oma eesmärgile soovitud tegevustega jätkata