TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Hendrik Mae IAIB164370

TOIDUKOHTADE TELLIMUSTE ESITAMISE KESKKONNA HALDUSLIIDESE VEEBILAHENDUSE ARENDUS

Bakalaureusetöö

Juhendaja: Inna Švartsman
MSc

Tallinn 2020

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 Hendrik Mae

17.05.2020

Annotatsioon

Toidukohtade tellimuste esitamise keskkonna haldusliidese veebilahenduse arendus

Käesoleva lõputöö eesmärk on luua restoranidele funktsionaalne osa terviklahendusest, millega kliendid lauast tellimusi esitada saavad. Välja arendatakse lahenduse haldusliides, kus restorani töötaja saab menüüd redigeerida ja olukorda saalis reaalajas jälgida.

Töö käigus analüüsitakse olukorda turul. Seejärel viiakse läbi küsitlused teenindusvaldkonna töötajatega, et panna paika rakenduse funktsionaalsed nõuded.

Analüüsi põhjal arendatakse välja veebirakenduse MVP, millega viiakse läbi kasutusmugavuse testid, et leida disaini käigus tekkinud vead.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 6 peatükki, 10 joonist, 4 tabelit.

Abstract

Developing a back office for restaurant order placing environment

The purpose of this thesis is to develop a functional part of a complete solution for restaurants, which allows their customers to make orders from the table.

To achieve this result, firstly a business analysis is conducted to study similar and competing products. Afterwards service sector employees will be surveyd for the purpose of listing out functional requirments of the application.

A MVP of this application will be developed based on the results of the analysis. This will be user tested to find design flaws made in the design process.

The thesis is in Estonian and contains 26 pages of text, 6 chapters, 10 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*, rakendusliides
Angular	Kasutajaliideste arendamiseks loodud Javascripti teek
Backend	*Frontendi* teenindav rakendus serveris
Back office	Rakendust toetav haldussüsteem
BEM	*Block element modifier*, stiili klasside nimetamise konventsioon
Chatbot	Arvutiprogramm, mille eesmärk on simuleerida vestlust inimkasutajatega
CSS	*Cascading style sheets*, keel veebi disainimiseks
Deploy	Serverisse paigutamine
EC2	Amazoni poolt pakutav serveriruumi rentimise teenus
Express	Javascriptile ehitatud teek, mis aitab REST teenuseid luua
Frontend	Kasutajale nähtav osa rakendusest
Git	Versioonihaldustarkvara
HTTP	*Hypettext Transfer Protocol*, hüperteksti edastusprotokoll
Javascript	Objektorienteeritud programmeerimiskeel
Json	*JavaScript Object Notation*, JavaScripti programmeerimiskeele alamhulgal põhinev andmevahetusvorming
MongoDB	NoSQL andmebaasimootor
Mongoose	Objekt-dokumendi teisendaja
MVC	*Model view controller*, rakenduse arhitektuuri mudel
MVP	*Minimal viable product*, minimaalne kliendile väärtust toov versioon tootest
Node.js	Javascripti jooksutamise keskkond
NoSQL	Mitterelatsiooniline andmebaasisüsteem
React	Kasutajaliideste arendamiseks loodud Javascripti teek
Redux	Oleku haldamise teek veebirakenustele
REST	Veebirakenduste arhitektuur
Sass	CSS eelprotsessor

SCP	*Secure copy*, andmete kopeerimine masinate vahel kasutades SSHga sama protokolli
Shell skript	Arvuti programm
Socket.io	Javascripti teek reaalajas suhtlevate rakenduste jaoks
SSH	*Secure Shell*, krüptograafiline võrguprotokoll masinate vahel suhtelsiseks
S3	Amazoni poolt pakutav failide talletamise pilveteenus
UML	*Unified modeling language*, ühtne modelleerimiskeel
Vue	Kasutajaliideste arendamiseks loodud Javascripti teek

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Motivatsioon ja probleem

Restoranides käib tänapäeval teenindus läbi teenindajate. Nad võtavad klientidelt vastu tellimuse ja edastavad selle kööki. Söögi valmides viivad nad selle lauda ja lõpuks võtavad kliendilt raha. Seda protsessi, nagu ka paljusid teisi, on hakatud infotehnoloogia arenedes uute tehniliste lahendustega optimiseerima.

Toitlustusvaldkonnas on esimesi innovatsioone läbi viinud juba O'Learys spordibaar. Seal on laudadele fikseeritud tahvelarvutid, kust klient saab ilma teenindaja sekkumiseta toidu lauda tellida. Selline lahendus aitab tõenäoliselt alandada personalikulusid vähendades korraga tööl olevate teenindajate arvu. Lisaks vähendab see teenindajate poolt tehtavaid vigu tellimuste üles kirjutamisel. O'Learyse poolt kasutusele võetud protsessi juurutamine on aga kulukas, sest nõuab suures koguses tahvelarvutite olemasolu.

Probleemi lahendamiseks luuakse veebirakendus, kust toitlustusasutuse klient saab isiklikust telefonist menüüs olevaid tooteid tellida ja vajaduse korral klienditeenindajat lauda kutsuda.

1.2 Töö eesmärk

Käesoleva töö eesmärk on luua eelmainitud lahenduse *back office* MVP ja viia läbi esmane kasutajatestimine.

MVP on uue turule toodava toote versioon, mis lubab meeskonnal vähima vaevaga koguda klientidelt maksimaalselt palju tagasisidet. MVP peamine kasu on arusaam klientide huvist toote kohta ilma seda täielikult välja arendmata. Mida varem saab valideerida huvi toote vastu seda vähem aega ja ressursse kulub reaalse huvi puudumise korral. [1]

MVP raames peab rakendus võimaldama restorani töötajatel redigeerida klientidele nähtavat menüüd ja mugavalt hallata klientide poolt läbi rakenduse tehtud kutsungeid.

1.3 Töö käik

Nii lõppkliendile kui ka restorani töötajatele nähtav keskkond valmisid samal ajaperioodil.

Enne arendust viidi läbi turuolukorra analüüs. Samuti tehti toitlustusvaldkonnas töötavate osapooltega intervjuud, et paika panna funktsionaalsed nõuded.

Seejärel loodi rakenduse peamiste vaadete kohta makett, mille abiga sai ideid täiendavalt valideerida. Maketi põhjal koostati rakendusele vaated ja seoti serveri rakenduse abiga klientide vaatega. Esimene arenduse iteratsioon valideeriti kasutusmugavuse testimisega.

2 Ärianalüüs

Käesolevas peatükis võetakse kokku läbi viidud analüüs eesmärgiga panna paika rakenduse funktsionaalsed nõuded. Selleks viidi läbi intervjuu restorani omaniku ja teenindajaga.

2.1 Olukord turul

Selleks, et võtta eeskuju juba eksisteerivatest lahendustest, millel on olnud aega turu vajaduste ja omapärade järgi areneda, analüüsiti nende häid ja halbu külgi.

2.1.1 Konkurendid

Ordly on otsene konkurent, kes pakub sarnast lahendust. Ordlyl on palju lisafunktsionaalsuseid, nagu laudade bronneerimine, eeltellimused ja Facebooki *chatbot*. Samas pole nende teenindamise voog täielikult läbi mõeldud. Näiteks teenindaja kutsumisel pole võimalik valida kas soovitakse maksta või lihtsalt teenindust. Selle tõttu võib tekkida olukord, kus teenindaja saab alles kliendi juurde jõudes teada, et ta oleks pidanud kaarditeminali kassa võtma, raisates selle hilje toomisega nii teenindaja kui ka kliendi aega.

Toote arenduse alguses otsustati, et rakendus läheb süvitsi ühe probleemi kasutusmugavuse parendamisega. Nii saab paremini panustada selle osa kvaliteedile.

2.1.2 Võrreldavad tooted

Kõige sarnasema lahendusega rakendused on Wolt ja Bolt Food. Neil käib tellimuste esitamine samuti läbi kliendi telefoni. Selleks, et tellimus jõuaks kassasüsteemi ja seejärel kööki saadavad nad läbi rakenduse tehtud tellimused kassasse paigutatud tahvelarvutitele. Sealt võtavad teenindajad tellimuse vastu ja kannavad kassasüsteemi. Kui tellimust täita ei saa, lükatakse see teenindaja poolt tagasi. Peale tulutuid vestluseid kassasüteemi CompuCash pakkujaga, et luua integratsioon nende süsteemiga, otsustati kasutada Wolt ja Bolt Foodiga sarnast lahendust.

Menüüde haldamine käib eelnevalt mainitud süsteemides vastavate teenusepakkujate tagarakendusestest eraldi. Kui menüüsse tuleb muudatus, siis tuleb see sisse kanda nii kassasüsteemis kui ka igas välise teenuse pakkuja rakenduses.

Ektaco pakub oma kassasüsteemiga kaasa lisateenuseid CompuCash Waiter ja CompuCash POS. Need on mobiilsetele seadmetele arendatud rakendused, mis ühenduvad Ektaco enda kassasüsteemiga. Tegemist on töös arendatava lahendusega sarnaneva tootega, mis erineb selle poolest, et tellimusi sisestavad süsteemi klientide asemel teenindajad.

2.2 Restorani siseelu

Kuna rakendust kasutaksid lisaks restorani klientidele ka klienditeenindajad siis oli oluline mõista nende tööharjumisi, et nende elu võimalikult lihtsaks teha.

2.2.1 Klientide teenindamine

Teenindajatele jäävad üldjuhul meelde saalis paiknevate laudade numbrid. Algne hüpotees oli, et teadete nimekirja kuvamisest piisaks saalis toimva kohta info teenindajatele edasmiseks. Konsulteerides hetkel töötava klienditeenindajaga jõuti selgusele, et loodavas rakenduses peab olema visuaalne kaart.

Kiirematel aegadel ei jõua klienditeenindaja minna leti taha keskse paneeli juurde laudade seisu kontrollima. Selleks, et kutsed teenindamata ei jääks, peab võimaldama teenindajatel laudade seisu ka saalis olles telefonist vaadata.

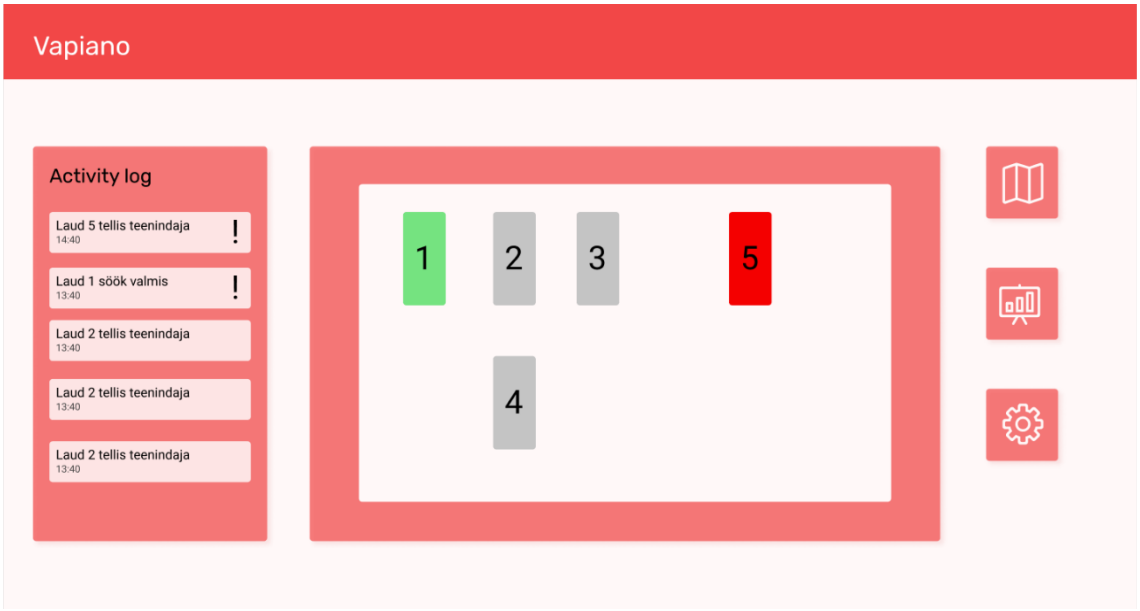
Kuna tegemist on MVPga millega ideed kõigepealt valideerida siis otsustati algselt läbi rakenduse maksmist mitte implementeerida.

2.3 Disain

Veebimakketide loomine on viis veebisaidi teenuse struktuuri ja funktsionaalsuse kavandamiseks lehele enne visuaalse disaini ja sisu lisamist. Sellega saab kiirelt luua ja muuta visandeid potesnsiaalsete kasutuslugude ja sisu paigutuse kohta. [2]

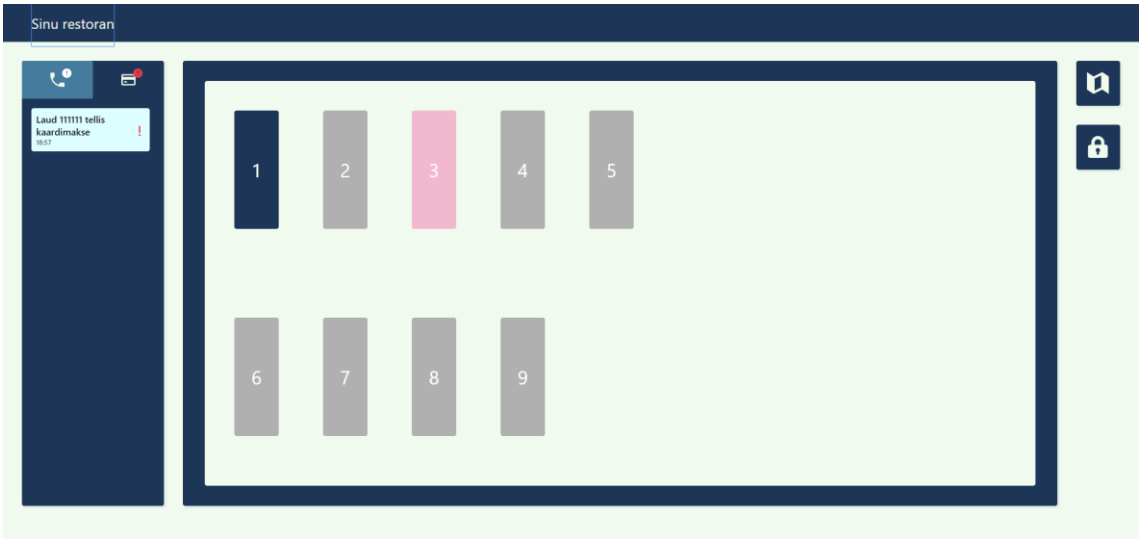
Rakenduse disainimseks ja ideede valideerimiseks loodi veebimaketid Figmas. Seejärel valideeriti disainid restorani töötajaga.

Joonisel 1 on välja toodud ühe kuva maketi lõppversioon, millega valideeriti restorani põrandaplaani visuaalse näitamise vajadust. Joonisel 2 on kujutatud kuva lõplik realisatsioon.



Joonis 1. Figmas loodud veebimakett

Joonisel 2 on ekraanikuva reaalsest rakenduse implementatsioonist, mis sai joonisel 1 kujutatud maketi järgi loodud.



Joonis 2. Maketi järgi realiseeritud kuva

3 Tehnoloogiad

Käesolevas peatükis antakse ülevaade arendamisel kasutatud keeltest, raamistikest ja tehnoloogiatest.

3.1 Eesrakendus

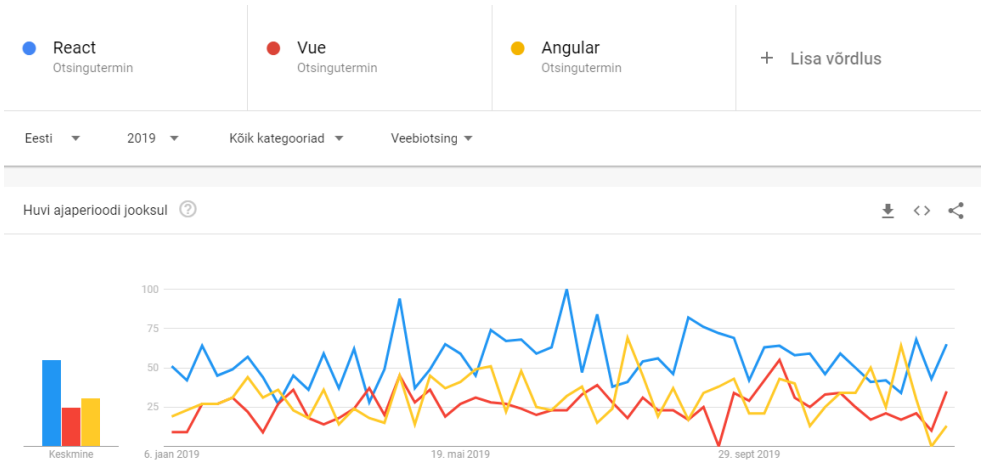
Platformiks, millel oma rakendust pakkuda sai valitud veebilehitsejad. Eesmärgiks oli võimalikult suur ligipääsetavus potensiaalsetele klientidele. Teenust pidi saama kasutama restoranis juba olemasolevatel seadmetel. Sarnase paindlikuse saavutamiseks peaks muidu tegema eraldi arendustöö Androidile ja IOSile.

3.1.1 React

React on Javascripti teek, mis on loodud kasutajaliideste arendamiseks. [3]

Frontendi arendamiseks sai valitud React. Variandid, mida autorid kaalusid olid Angular, Vue ja React. Kuna antud tööd on võrdselt hästi võimalik teha kõikide raamistikega siis autorid otsustasid Eesti näiliselt kõige populaarsema raamistiku kasuks.

Joonis 3 näitab Google Trendsist võetud graafikut kolme eelnevalt mainitud raamistiku otsingute tiheduse kohta, kus x-teljel on kuvatud kuupäev ja y-teljel on Google otsingute arv mis antud terminiga tehti.



Joonis 3. Veebiraamistike populaarsus aastal 2019 [4]

3.1.2 Redux

Redux on JavaScripti rakendustele loodud oleku haldamise konteiner. See aitab kirjutada rakendusi, mis käituvad järjepidevalt, töötavad erinevates keskkondades ning on lihtsalt testitavad. [5]

Redux sobib hästi olukordades kus keerulistes komponentide hierarhiates on vaja rakenduse olekut kasutada ja muuta. See pakub lahendust kahele probleemile, mis suurtemate komponentpõhiste veebirakenduste arendamisel esineb.

Esiteks kuidas ühes komponendis andmete muutumisel teist komponenti sellest teavitada ja teiseks info edastamine läbi mitme komponendi kihi. Sellest räägitakse täpsemalt rakenduse struktuuri alampeatükis „Tehnilised lahendused“.

3.1.3 Sass

Sass on CSS stiililehtede eelprotsessor. See võimaldab muutujate, funktsioonide ja pesastatud reeglitega luua lihtsamini hallatavat koodi. [6]

Sass valiti, et koodi paremini organiseerida ja seeläbi hallatavamaks teha. Reeglite pesastamine võimaldab stiilide koodi kirjutamisel kasutada Block Element Modifier (BEM) metodoloogiat. BEM on lihtne ja võimas nimetamise konventsioon mis muudab *frontend* koodi loetavamaks, paremini skaleeruvaks, robustsemaks ja rangemaks. [7]

Muutujad võimaldavad arenduse käigus rakenduse kiiremat stiliseerimist ning võimaluse luua tulevikus kasutajale rakenduse sisene värvipaleti muutmine.

3.2 Tagarakendus

Käesolevas peatükis antakse ülevaade rakenduse *backend* arendamisel kasutatud tehnoloogiatest.

3.2.1 Node.js

Node.js on avatud lähtekoodiga asünkroonne Javascripti käitusaja keskkond. See jookseb võimeka Google V8 Javascripti mootori peal. [8]

Kui Node.js rakendus peab tegema sisend- või väljundtoimingu, jätkab Node.js teiste toimingutega, selle asemel et lõime blokeerida ja raisata protsessori tsükleid ootamise peale. See teeb samaaegsete ühenduste teenindamise kiiremaks. [8]

Olukorras kus üheaegselt on vaja teenindada mitmeid kliente erinevates restoranides, sobib Node.js asünkroonne struktuur ideaalselt ja sellest lähtuti valiku tegemisel.

3.2.2 Express

Express on minimaalne ja paindlik Node.js veebirakenduste raamistik, mis toetab veebi- ja mobiilirakenduste loomist. See pakub mitmeid HTTP teenuseid ja vahevara, tänu millele on API-liidese loomine kiire ja lihtne. [9]

Lisaks Expressile on Node.js jaoks loodud teisigi raamistikke, mis teenuste arendamist hõlpsustavad. Alternatiivdeks on näiteks Sails.js ja Fastify. Sails.js lubab rakenduste ehitamist veelgi kiiremaks muuta ja Fastify on loodud suurema jõudlusega alternatiivina.

Express valiti, kuna see on kõige vanem ja sellel on ajaga välja arenenud hea ökosüsteem, millele õppimisel ja rakenduse silumisel toetuda saab. Üheks näiteks on järgmises peatükis viidatud Socket.io dokumentatsioon, mille näited on kirjutatud just Expressi näitel.

3.2.3 Socket.io

Socket.IO on Node.jsile loodud teek kahesuunalise suhtluskanali loomiseks kliendi ja serveri vahel. See tähendab, et server saab klientidele sõnumeid saata. Kui serverile saadetakse sõnum, mille tagajärjel on vaja teisi kuulajaid muudatustest serveris teada anda, siis edastab server selle sõnumi edasi kõigile teistele ühendatud klientidele. [10]

Sokkeli kasutamine rakenduses on vajalik, et reaalaajas teavitada teenindajaid klientide tehtud kutsetest.

3.3 Baasikiht

Käesolevas peatükis antakse ülevaade andmebaasi kihis kasutatud tehnoloogiatest.

3.3.1 MongoDB

MongoDB on dokumendiandmebaas, mis tähendab, et see hoiab andmeid json-kujulistes dokumentides. [11]

Andmebaasiks sai valitud MongoDB. Ennustades vajadust andmeid erinevate restoranide ja kassasüsteemide järgi kohandada oli NoSQL õige otsus. Selle dünaamiline andmestruktuur lubab tulevikus kiirelt kohandada rakendust erinevate menüü eripärade järgi, nagu allahindlused, alamkategooriate lisandumised või variatsioonide esinemised toodetes.

3.3.2 Mongoose

Mongoose on objektidokumendi modelleerimise kiht, mis on ehitatud Node.js MongoDB draiveri peale. See sarnaneb relatsiooniliste andmebaaside jaoks loodud objektrelatsiooni modelleerijatega. [12]

Mongoose lihtsustab rakenduse arendamist, andmemudelite defineerimise ja andmete töötlemisel nende valideerimsiga.

3.4 Taristu

Käesolevas peatükis antakse ülevaade rakenduse jaoks kasutusele võetud taristust.

3.4.1 Amazon EC2

Amazon EC2 pakub Amazoni veebiteenuste pilves skaleeritavat serveriruumi. Amazon EC2 kasutamine kõrvaldab algse vajaduse investeerida riistvarasse, kiirendades rakenduste arendust ja juurutamist. Amazon EC2 võimaldab kasutatavaid ressursse teenuse nõudluse muutumisel lihtsalt skaleerida, vähendades liikluse prognoosimise vajadust. [13]

Amazon EC2 sai valitud selle töökindluse ja kasutusumugavuse tõttu. Lisaks oli autoril aastane prooviperiood. See on piisav aeg, et MVPga idee valideerida.

3.4.2 Amazon S3

Amazon Simple Storage Service on loodud selleks, et muuta veebipõhine andmete tallemine arendajatele lihtsamaks.

Amazon S3-l on lihtne veebiteenuste liides, mida saab kasutada andmete talletamiseks ja allalaadimiseks. See annab juurdepääsu samale väga skaleeritavale, usaldusväärsele, kiirele ja odavale andmesalvestusinfrastruktuurile, mida Amazon kasutab oma globaalse veebisaitide võrgu käitamiseks. Teenuse eesmärk on maksimeerida mastaabisoodustusi ja anda need eelised arendajatele edasi. [14]

Amazon S3 valiti samuti töökindluse ja kasutusmugavuse tõttu.

3.4.3 Serverisse paigutamine

Selleks, et rakendus serverisse *deployda*, loodi shell skriptid.

Veebilehtede deploymiseks loodi skript, mis kasutab serveri privaatvõtit, et Linuxi scp käsklusega kood üle võrgu õigesse kausta kopeerida.

Node.js rakenduse deploymiseks loodi skript, mis ühendab SSHga Amazoni EC2 serverisse, tõmbab lähtekoodi Gitist alla, sisestab lähtekoodi andmebaasi ja S3 paroolid ning käivitab rakenduse.

4 Rakenduse struktuur

Käesolevas peatükis kirjeldatakse rakenduse kasutajale nähtavat struktuuri, andmemudelit ja tehnilisi lahendusi.

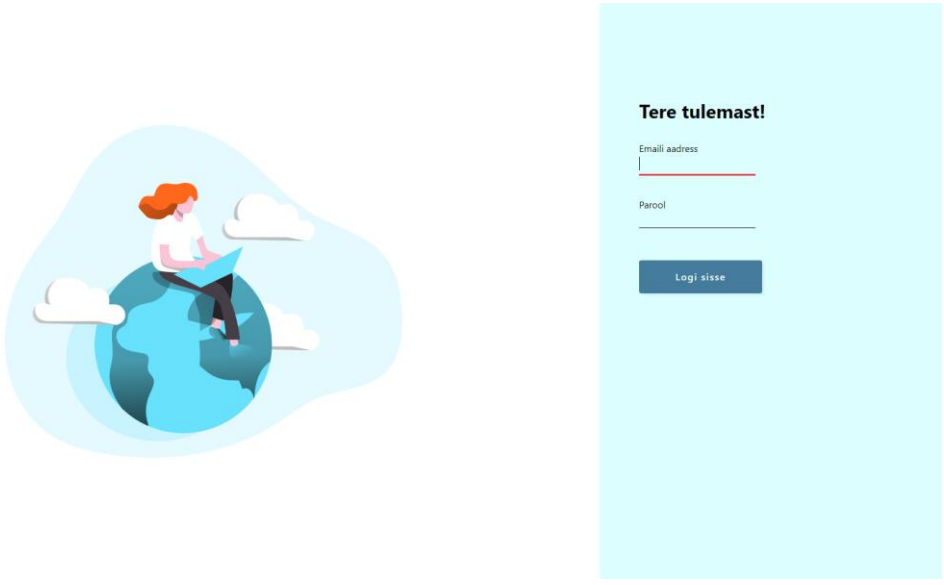
4.1 Rakenduse struktuur

Rakendus koosneb sisse logimise kuvast ja kahest suuremast moodulist, mis koos sisu muutuste ja hüpikakendega kasutaja voogusid juhivad.

Lähtudes ärianalüüsi peatükis välja toodud vajadustest reageerib rakendus seadme kuvasuurusele. Mobiili vaates otsustati loobuda menüü redigeerimise võimalusest.

4.1.1 Sisse logimse kuva

Joonisel 4 on kujutatud sisselogimise kuva



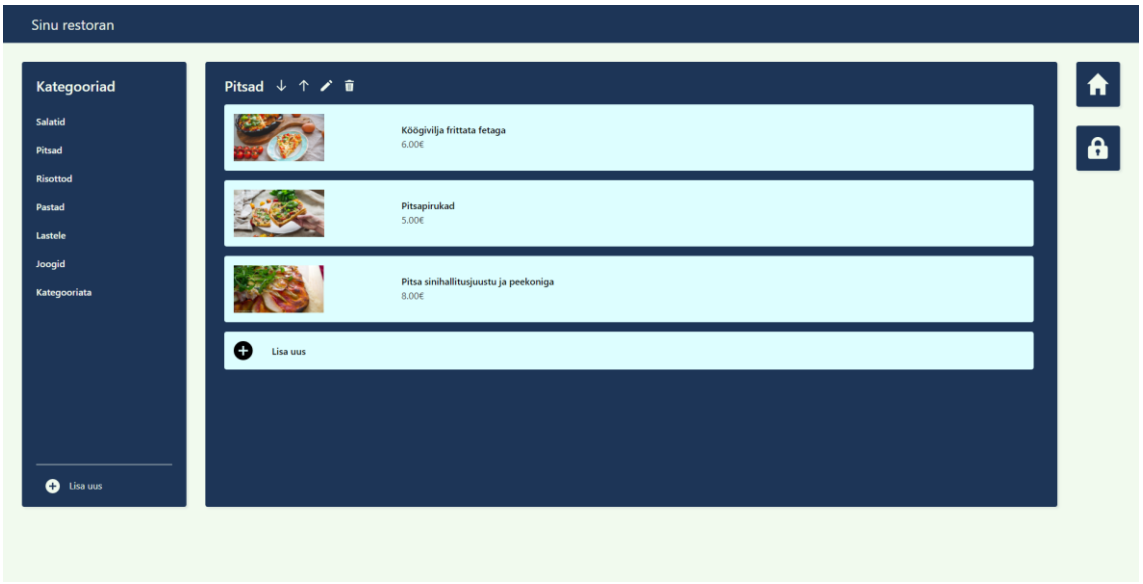
Joonis 4. Rakenduse sisse logimise kuva

4.1.2 Menüü haldamise moodul

Menüü haldamise moodulis saab läbi viia toiminguid kategooriate ja müüdavate toodetega.

Kategooriaid saab lisada, kustutada, ümber nimetada ja järjestuses ümber paigutada.

Joonisel 5 on kujutatud menüü haldamise mooduli kuvatõmmis menüü sirvimise olekus.



Joonis 5. Rakenduse menüü kuva

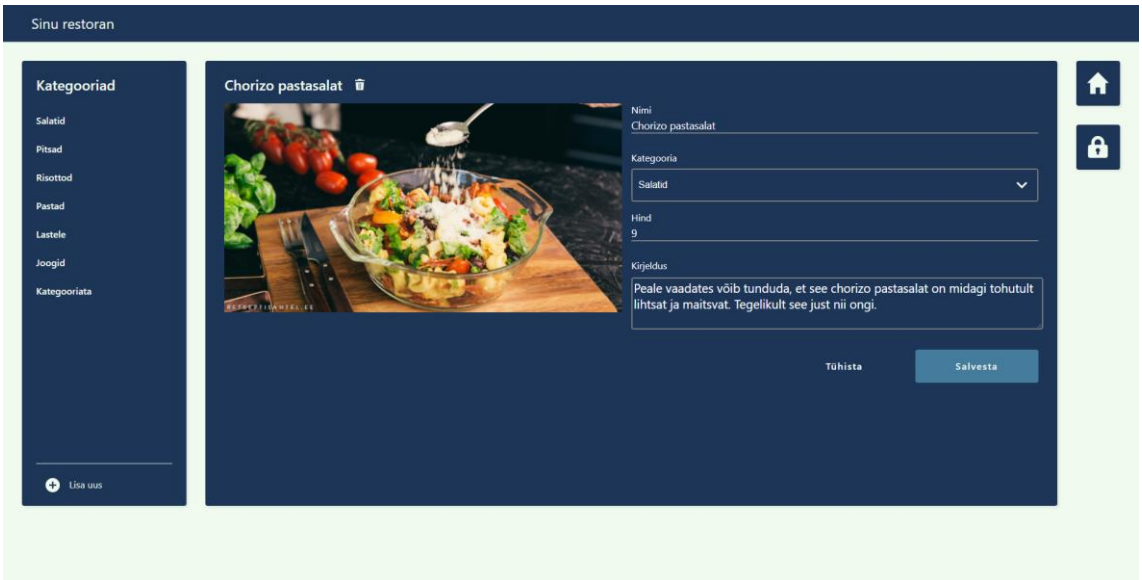
Kuva paremas ääres on navigatsiooni tulp.

Kuva vasakus ääres on kategooriate paneel, mille jalutsis on uue kategooria lisamise nupp, mis avab vajutusel hüpikmenüü, mis küsib uue kategooria nime.

Kuva keskel on ühte kategooriasse kuuluvate toodete nimistu paneel. Selle päises on valitud kategooria nimi koos tegevustega, mis selle kategooriaga teha saab. Kategooria nimistu viimane rida on uue toote lisamise nupp.

Tooteid saab lisada, kustutada ja muuta.

Joonisel 6 on kujutatud menüü haldamise mooduli kuvatõmmis toote redigeerimse olekus.



Joonis 6. Rakenduse toote muutmise kuva

Redigeerimise olekus on keskmisel paneelil paremas tulbas vorm eseme omaduste kohta ja vasakus tulbas pilt tootest.

Pildile klikkides avaneb hüpikaken, kust kaudu saab arvutist tootele uue pilti üles laadida. Paneeli päises on redigeeritava toote nimi või pealkiri toote lisamise kohta ja nupp, millest olemasolevat toodet kustutada saab.

Tabelis 1 on välja toodud kategooria sisemine struktuur.

Välja nimi	Andmetüüp	Kohustuslik väli	Selgitus
_id	Number	+	Primaarvõti
name	String	+	Kategooria nimi
order	String	+	Kategooria positsioon nimistus
organizationId	Reference	+	Viide asutusele

Tabel 1. Kategooriate sisemine andmestruktuur

Tabelis 2 on välja toodud müüdava toote sisemine struktuur.

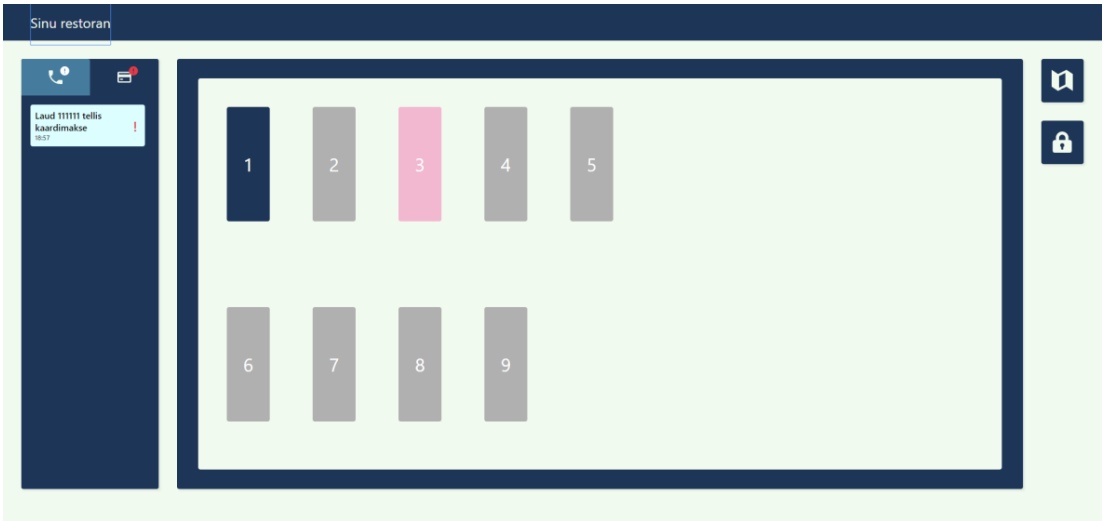
Välja nimi	Andmetüüp	Kohustuslik väli	Selgitus
_id	Number	+	Primaarvõti
title	String	+	Toote nimetus
description	String	-	Toote kirjeldus
price	Number	+	Toote hind
imageUrl	String	-	Viide toote pildile
categoryId	Reference	+	Viide toote kategooriale
organizationId	Reference	+	Viide asutusele

Tabel 2. Toodete sisemine andmestruktuur

4.1.3 Kutsungite haldamise moodul

Kutsungid laekuvad rakendusse reaalajas. Eristama peab tellimust, teenindaja kutsungit tellimiseks ja teenindaja kutsumist maksmiseks.

Joonisel 7 on kujutatud kutsungite haldamise mooduli arvuti ja tahvelarvuti vaate ja joonisel 8 on kujutatud moobilivaate kuvatõmmis.



Joonis 7. Rakenduse tellimuste haldamise tahvelarvuti kuva

Joonis 8. Rakenduse tellimuste haldamise mobiili kuva

Tabelis 3 on välja toodud teenindaja kutsungi sisemine andmestruktuur.

Välja nimi	Andmetüüp	Kohustuslik väli	Selgitus
_id	Number	+	Primaarvõti
callType	String	+	Kutsungi tüüp
isWaiting	Boolean	+	Kas kutsung vajab tähelepanu
createdTime	String	+	Aeg, millal kutsung tehti
createdDate	String	+	Kuupäev, millal kutsung tehti
tableId	Reference	+	Viide lauale

Tabel 3. Teenindaja kutsungi sisemine andmestruktuur

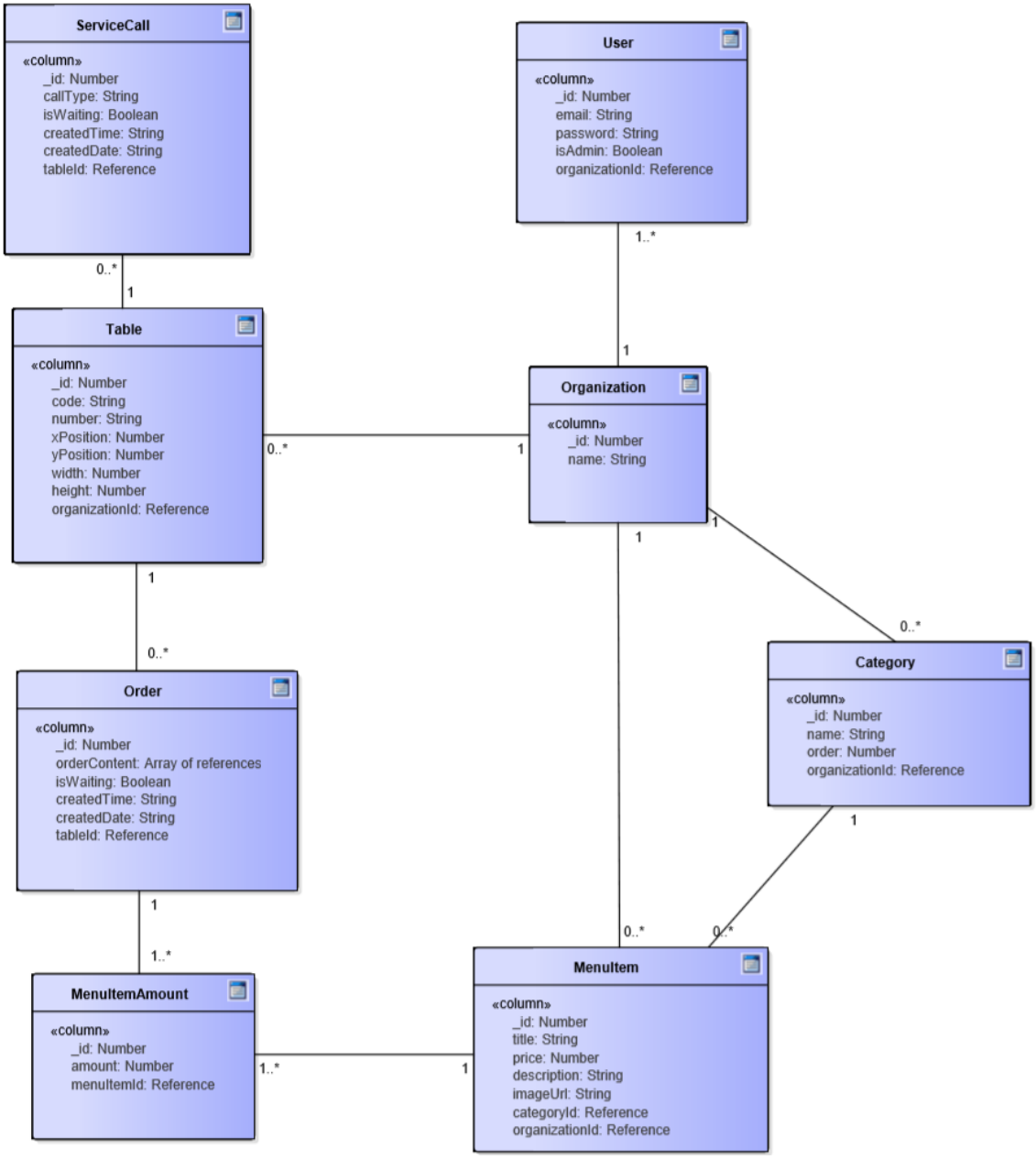
Tabelis 4 on välja toodud tellimuse sisemine andmestruktuur.

Välja nimi	Andmetüüp	Kohustuslik väli	Selgitus
_id	Number	+	Primaarvõti
orderContent	List of references	+	Tellimuse sisu
isWaiting	Boolean	+	Kas tellimus vajab tähelepanu
createdTime	String	+	Aeg, millal kutsung tehti
createdDate	String	+	Kuupäev, millal kutsung tehti
tableId	Reference	+	Viide lauale

Tabel 4. Tellimuse sisemine andmstruktuur

4.2 Andmebaas

Joonisel 9 on lisaks toodud UML graafik, et paremini edasi anda andmete omavahelisi seoseid.



Joonis 9. Rakenduse andmestruktuuri UML joonis

4.3 Tehnilised struktuurilahendused

4.3.1 Backend struktuur

Backendi struktureerimisel kasutati MVC arhitektuuri.

MVC arhitektuuri peamine mõte on, et igal koodilõigul on oma eesmärk. See aitab koodi põhifunktsioonid korrastada omaette kastidesse. Mudeli kiht koodist sisaldab rakenduse andmeid, teenusekiht tegeleb äriloogikaga ja kontrolleri kiht koodist võtab vastu päringuid. [15]

Tänu koodi funktsioonipõhisele eraldamisele saab neid taaskasutada. Näiteks teenuskihi funktsioone sai kasutatud nii REST päringute kui ka sokkelite teenindamiseks.

4.3.2 Frontend struktuur

Rakenduse erinevate lehtede vahel navigeerimiseks kasutati React Routerit.

React Router on Reactile loodud API mis aitab luua üheleherakendusi. Selliste rakenduste marsruutimine toimub koodi, mitte serveri tasandil. See lubab luua kiiremaid ja sujuvamaid kasutajakogemusi, eemaldades navigeerimisel lehe uuesti laadimised ja vähendades infomahtu, mida lehitseja igal navigeerimisel alla laadima peab. [16]

Rakenduse olekute hoidmiseks ja juhtimiseks kasutati Reduxit. Redux pakub lahendust kahele probleemile, mis suurtemate komponentpõhiste veebirakenduste arendamisel esineb.

Esimene probleem, mida Redux lahendab, on andmete puurimine. See tähendab andmete komponentide puus mööda mitut sõlme alla liigutamist. Andmete puurimine on problemaatiline, sest igale komponendile paljude erinevate andmete kaasa andmisel muutub koodi haldamine tülikamaks ja sellest aru saamine raskemaks. Lisaks on sellist koodi raske siluda, sest andmete liikumist on raske jälgida. Antud töös oleks selline olukord ette tulnud sisse logimisel saadud kasutaja andmete kõikidesse komponentidesse toimetamisel. Selle asemel salvestatakse andmed Reduxi konteinerisse ja olekut vajavad komponendid saavad andmed sealt kätte.

Teine probleem, mida Redux lahendab on rakenduse oleku haldamine. Selleks, et realiseerida terve rakenduse poolt kasutatavaid komponente nagu hüpikmenüüsid ja aknaid, paigutati need väljaspoole ruuteri poolt kuvatavaid komponente. Neile loodi Reduxi konteiner, mis hoiab nende ekraanil kuvamise tõeväärtust ja sisu. Seejärel anti hüpikaknaid kuvavatele komponentidele ligipääs sellele Redux konteinerile.

4.4 Integratsioonitestid

Kuna arendatud rakenduses on äriloogika kiht väga õhuke, siis otsustati unit testide asemel integratsioonitestide kasuks.

Integratsioonitestimine on tarkvaratestimise vorm, kus testitakse rakenduse eraldiseisvate osade koos töötamist. Selle eesmärk on kõrvaldada rikked integreeritud üksuste omavahelises suhtluses. [17]

Integratsioonitestide kirjutamiseks loodi eraldiseisev Java rakendus. Testid kirjutati kasutades Cucumber ja Selenium teeke. Testimiseks kasutati „Valideerimine“ peatükis välja toodud kasutuslugusid.

Selenium on mitmesuguste veebibrauserite automatiseerimist võimaldavate tööriistade ja teekide projekt. See lubab emuleerida kasutajate tegevusi veebibrauserites. [18]

Cucumber on testimise tööriist mis lubab kirjutada teste, millest igaüks, olenemata tehnilisest taustast aru saab. Selle jaoks kasutatakse Gherkin keelt [19]

Järgmisena on esitatud näide Cucumber testist kirjutatud Gherkin keelega.

```
Given I am using a calculator
And I enter "10" as number 1
And I enter "5" as number 2
When I press the calculate button
Then I should see "15" as output
```

Joonis 10. Gherkin keeles kirjutatud testi näide

5 Valideerimine

Käesolevas peatükis kirejaldatakse rakenduse esimese testimise ringi käiku ning tulemusi.

5.1 Kasutusmugavuse testimise taust

Kasutusmugavuse testimine on veebilehe, rakenduse või muu digitaalse toote katsetamise meetod, millega jälgitakse inimesi üritamas ülesandeid täita. Selle eesmärk on välja tuua kasutamise käigus tekkivaid segadusi ja võimalusi toote parendamiseks, et kasutajakogemust parandada.

Kasutusmugavuse testimist tehakse päris inimestega, kes märkavad projekti kallal töötajatest suurema tõenäosusega vigasid. Seda seetõtu, et toote arendajad on pikalt projektiga tegelenud ja sellega juba tuttavad. Tihtipeale pimestab toote sügavuti tundmine disainereid, turundjaid ja tooteomanikke veebilehe probleemide ees.

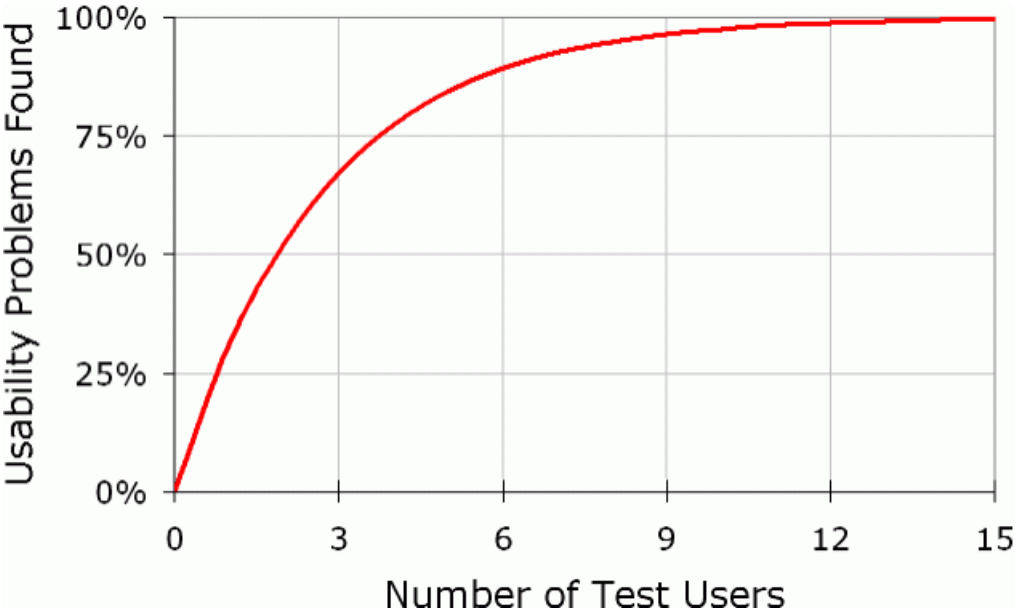
Seda tüüpi uuringud on eriti olulised uute toodete või disaini uuendustega. Ilma teiste inimestega testimiseta, võib jääda kinni protsessidesse ja ideedesse, mida meeskonna liikmed mõistavad, kuid sihtgrupp mitte. [20]

Töö autor valis testgrupi suuruseks viis inimest. Valik sai tehtud lähtudes asjaolust, et mida rohkem kasutajaid testida, seda vähem informatsiooni saab igalt järgnevalt testitavalt.

Testides ühte inimest, on kogu saadud info uus. Testides kahte inimest, leiab suure tõenäosusega uusi vigu, kuid korduvad ka vanad. Mida rohkem testitavaid lisada, seda vähem uusi vigu iga uus testitav leiab.

Kui minna selle mõttekäiguga ekstreemsusesse, siis võiks argumenteerida, et ühe isikuga testid annaksid kõige parema tulemuse. See ei pea paika, sest üksik testitav võib katse käigus teha kogemata tavapäratuid valikuid. Sellised üksikud juhtumid ei esinda suuremat gruppi. Mida rohkem inimesi testida, seda rohkem väheneb juhuslikkuse mõju testile. [21]

Joonisel 11 on kujutatud graafik mis näitab logaritmilist sõltuvust testitavate ja leitud probleemide vahel.



Joonis 11. Grafik kasutajatestimisel leitud vigade suhtest testijate arvuga [21]

5.2 Kasutusmugavuse testimise läbiviimine

Selleks, et esimeses arenduse-testimise ringis ressursse kokku hoida, otsustati testimida restorani tööga mitte kokkupuutunud inimestega. Nii leiti üles erialaseid teadmisi mittevajavad kasutusmugavuse probleemid, et need enne valdkonnaga seotud inimestega testimist ära parandada. Niimoodi saab teadlikumate kasutajate kogemused sisulisemate probleemide uurimiseks jätta.

Kuna teenindajaks võib minna igaüks, sest teenindaja ameti jaoks üldiselt pole vaja varasemat treeningut, ega haridust,[22] otsustati testitavateks valida erinevate taustadega juhuslikult inimesed, kes polnud varem rakendust näinud.

Testimine toimus läbi videokõne nii, et testitavad jagasid oma ekraani pilti. Enne testi anti neile süsteemi kasutaja meiliaadress ja parool. Esmasel sisselogimisel paluti neil kirjeldada mida nad peakuval näevad ja kuidas nad kuvatud komponente enda jaoks intuitiivselt tõlgendavad. Pärast lasti neil läbi teha ette valmsitatud kasutuslood ja lõpuks küsiti täiendavaid küsimus.

Tellimuste esitamise keskkonna *back office*'i kasutusmugavuse sooritatud kasutuslood olid järgnevad:

- Sisse logimine
- Menüüsse uue kategooria lisamine
- Menüüsse uue eseme lisamine
- Menüüs eseme info muutmine
- Kategooria nime muutmine
- Menüüst kategooria kustutamine
- Kategooriata jäänud esemete ümber paigutamine
- Teenindaja kutse kinnitamine
- Tellimuse kinnitamine

Lisainfo saamiseks küsiti testitavatelt järgnevad küsimused:

- Mis pakkus kasutuse juures kõige rohkem raskusi?
- Mis meeldis disaini juures enam?
- Mida muudaksid disaini juures?

5.3 Kasutusmugavuse testide tulemused

Viiest testitjast neljaga läks kõik plaanipäraselt. Viimane testija hakkas lehel iga ülesande juures vigu otsima ja tänu temale sai ka paar äärmusjuhtumit üles leitud.

Kõik testitavad said koheselt aru, et esivaatel on kuvatud restorani plaan. Lisaks said kõik neile antud ülesannetega peale esmaseid arusaamatusi hakkama.

Sellegi poolest leiti palju vigu, millest enamusi ei suutnud töö autor ette näha. Leitud vead saab liigitada kahte kategooriasse - koodivead ja kasutusmugavuse vead.

5.3.1 Koodivead

Hinna sisestamise väljal tekkisid kasutaja sisendi kontrollimisega seotud vead – sisestada sai matemaatilist konstanti e ja miinus märki.

Üritades muuta kategooria nime tühjaks sõneks, ei kuvatud veateadet.

Teatud ekraanisuurustele ei skaleerunud rakendus oodatul kujul.

5.3.2 Kasutusmugavuse vead

Kõik testijad olid konsensusel, et navigatsiooniriba implementatsioon on halb. Kaks viiest arvasid nuppude tähenduse enne peale vajutamist ära. Üks testija arvas, et see on hea lahendus tahvelarvutile.

Tellimuste kuva nupud tellimuste ja kutsungite vahel valimiseks olid oma värvikasutusega ebaselged. Kolm testijat viiest ei saanud tänu sellele aru, kumb valik aktiivne on.

Menüü kuvas ei olnud selge, et keskmise komponendi päises asuvad kategooria muutmise nupud olid kuidagi kategooriaga seotud.

Kategooria kustutamisel polnud selge, et kustutatud kategoorias olnud esemed liikusid „Kategooriata“ valiku alla.

Aktiivse kutsungita lauda prooviti avada, et sellega seotud infot näha.

Hinna sisestamise väljale prooviti edutult sisestada tähti, hinda prooviti eraldada komaga ja prooviti panna lõppu valuuta märki.

6 Edasised arendused

Käesolevas peatükis on välja toodud autori poolt järgmise planeeritud tööd.

6.1 Parandused disainis

Enne järgmist kasutusmugavuse testimise ringi peab parandama kõik üleval mainitud vead. Kuna ekraani suurusest tingitud erinevused ei häirinud kasutusmugavust, siis seda enne kliendi leidmist ja kasutatavate seadmete täpsustamist ei muudeta. Peale teist kasutusmugavuse testimise ringi plaanitakse teha viimased viimistlused, et siis potensiaalsete klientide juurde toodet reklaamima minna.

6.2 Konfiguratsiooni moodul

Kui leida esimene klient, siis on planeeritud arendama hakata konfiguratsiooni moodulit. Sealt peab organistatsiooni peakasutaja saama lisada väiksemate õigustega kasutajaid, muuta restorani laudade plaani ja kliendile nähtava menüü välimust.

6.3 Menüü arendused

Esemete ajutiselt mitteaktiivseks tegemine ja soodustuste rakendamine teevad töö rakendusega veel mugavamaks. Selle funktsionaalsuse loomine ootab samuti idee eelnevat valideerimist.

6.4 Kassasüsteem

Reaalsete klientide tekkimisel tuleb analüüsida võimalusi siduda rakendust kliendi olemasoleva kassasüsteemiga. See eemaldaks potensiaalselt vajaduse teenindajal tellimusi süsteemi kanda. Lisaks saaks laoseisu järgi toodete saadavust kuvada.

Kokkuvõte

Bakalaureusetöö eesmärgiks oli luua restoranidele teenindamise effektiivsemaks muutmise rakenduse MVP ja selle kasutusmugavuse valideerimine, et saavutada kindlus potesnsiaalsete klientide juurde minemisel. Selle jaoks uuriti kõigepealt toilustussektori töötajatelt rakenduse vajalikkuse kohta. Kui oli kindlus, et ideel on potensiaali, siis uuriti konkurente ja analüüsiti äriloogikat, et panna kokku makett mille järgi arendama hakata.

MVP *back office* tehti töö lõpustaadiumites nähtavaks domeenil tellimus.com/admin.

Rakenduse kood on esitamise hetkel nähtav Githubi repositooriumis aadressil *https://github.com/nils-emil/chatRestoraunt/compare/develop*

Võrgus nähtavat rakendust kasutati kasutusmugavustestide läbiviimiseks. Autori jaoks oli see esimene kord selliseid teste läbi viia ja ta oli positiivselt üllatunud kui palju konstruktiivset kriitikat sel moel on võimalik saada. Selline testimine teeb esimesed kohtumised potensiaalsete klientidega kindlasti konstruktiivsemaks.

Töö käigus sai autor esmakordselt õppida tehnoloogiaid nagu NodeJs, Express, Mongoose ja Socket.io. Lisaks oli see esimene täissuuruses rakendus, mille autor nullist valmiduseni ehitada sai.

Kasutatud kirjandus

[1] „Minimum Viable Product (MVP),“ [Võrgumaterjal]. Saadaval: https://www.agilealliance.org/glossary/mvp [Kasutatud 14 mai 2020].

[2] „What is wireframing?,“ [Võrgumaterjal]. Saadaval: https://www.experienceux.co.uk/faqs/what-is-wireframing [Kasutatud 15 mai 2020].

[3] „React,“ [Võrgumaterjal]. Saadaval: https://reactjs.org/docs/getting-started [Kasutatud 15 mai 2020].

[4] Google Trendsis genereeritud graafik, [Võrgumaterjal]. Saadaval: https://trends.google.com/trends/explore?cat=31&date=2019-04-30%202020-03-01&geo=EE&q=React,Angular,Vue [Kasutatud 30 jaanuar 2020].

[5] „Why use Redux? Reasons with clear examples,“ [Võrgumaterjal]. Saadaval: https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835 [Kasutatud 15 mai 2020].

[6] „Documentation,“ [Võrgumaterjal]. Saadaval: https://sass-lang.com/documentation [Kasutatud 14 mai 2020].

[7] „Bem,“ [Võrgumaterjal]. Saadaval: http://getbem.com [Kasutatud 15 mai 2020].

[8] „Introduction to Node.js,“ [Võrgumaterjal]. Saadaval: https://nodejs.dev [Kasutatud 16 mai 2020].

[9] „Express,“ [Võrgumaterjal]. Saadaval: https://expressjs.com [Kasutatud 16 mai 2020].

[10] „Socketio,“ [Võrgumaterjal]. Saadaval: https://socket.io/get-started/chat [Kasutatud 3 veebruar 2020].

[11] „MongoDB,“ [Võrgumaterjal]. Saadaval: https://www.mongodb.com [Kasutatud 16 mai 2020].

[12] „Top 4 Reasons to Use Mongoose with MongoDB,“ [Võrgumaterjal]. Saadaval: https://www.stackchief.com/blog/Top%204%20Reasons%20to%20Use%20Mongoose%20with%20MongoDB [Kasutatud 15 mai 2020].

[13] „What is Amazon EC2?,“ [Võrgumaterjal]. Saadaval: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts [Kasutatud 30 veebruar 2020].

[14] „What is Amazon S3?,“ [Võrgumaterjal]. Saadaval: https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome [Kasutatud 23 veebruar 2020].

[15] „MVC: Model, View, Controller - App organization explained,“ [Võrgumaterjal]. Saadaval: https://www.codecademy.com/articles/mvc [Kasutatud 3 veebruar 2020].

[16] „A Brief Overview of React Router and Client-Side Routing,“ [Võrgumaterjal]. Saadaval: https://medium.com/@marcellamaki/a-brief-overview-of-react-router-and-client-side-routing-70eb420e8cde [Kasutatud 16 mai 2020].

[17] „Integration Testing,“ [Võrgumaterjal]. Saadaval: http://softwaretestingfundamentals.com/integration-testing [Kasutatud 16 mai 2020].

[18] „The Selenium Browser Automation Project,“ [Võrgumaterjal]. Saadaval: https://www.selenium.dev/documentation/en [Kasutatud 16 mai 2020].

[19] „What is Cucumber Testing Tool? Framework Introduction,“ [Võrgumaterjal]. Saadaval: https://www.guru99.com/introduction-to-cucumber [Kasutatud 16 mai 2020].

[20] „A beginner's guide to usability testing,“ [Võrgumaterjal]. Saadaval: https://www.hotjar.com/usability-testing [Kasutatud 3 mai 2020].

[21] „Why You Only Need to Test with 5 Users,“ [Võrgumaterjal]. Saadaval: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users [Kasutatud 3 mai 2020].

[22] „Waiter or Waitress,“ [Võrgumaterjal]. Saadaval: https://www.truity.com/career-profile/waiter-or-waitress [Kasutatud 3 mai 2020].