TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Indrek Kann 185802IADB

Mobiilsetel seadmetel kasutamiseks ettenähtud restoranide veebirakenduse arendamine

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2022

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: Indrek Kann

04.01.2022

Annotatsioon

Käesoleva lõputöö eesmärk on luua mobiilsetel seadmetel kasutamiseks ettenähtud restoranide veebirakendus, mis lihtsustaks ja vähendaks teenindajate poolt tehtavat tööd ning parandaks rakenduse kasutajate külastuskogemust pakkudes neile kiiremat ja mugavamat teenust.

Käesolevas töös analüüsitakse erinevaid tehnoloogiaid, et valmiv rakendus realiseeritaks kasutades parimaid lahendusi. Suur fookus on rakenduse kiirusel ning heal kasutajakogemusel. Samuti on tähtis, et tulevikus funktsionaalsuse juurdelisamine oleks võimalik ilma suurema vaevata.

Töö käigus valmib kasutust leidev veebirakendus, millega on võimalus broneerida valitud restoranis laud, sirvida menüüd, lisada tooteid ostukorvi, toodete eest maksta ja soovi korral jätta jootraha.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 19 joonist, 1 tabelit.

Abstract

Development of a Web Application for Restaurants for use on Mobile Devices

The goal of this final thesis is to develop a web application for restaurants for use on mobile devices, that would simplify and reduce the amount of work done by waiters and would increase the user experience providing them with faster and more convenient service.

The thesis analyzes different technologies so that the application is implemented with the best solutions possible. The focus is on the performance of the application and user experience. It is also important, that the addition of future developments would be possible without much effort.

In the course of this thesis, a working and usable web application will be developed, with which it is possible to book a table in a selected restaurant, browse the menu, add products to the shopping cart, pay for the products and leave a tip if desired.

The thesis is in Estonian and contains 28 pages of text, 7 chapters, 19 figures, 1 table.

Lühendite ja mõistete sõnastik

.NET	Microsofti poolt arendatav avatud lähtekoodiga tarkvararaamistik
API	*Application programming interface*, rakendusliides
BFF	*Backends for frontends*, tarkvaraarenduse disainimuster
CLI	*Command-line interface*, käsurealiides
ERD	*Entity relationship diagram*, andmestruktuuri kirjeldav skeem
IOC	*Inversion of control*, programmeerimisprintsiip
ISO	*International Organization for Standardization*, Rahvusvaheline Standardimisorganisatsioon, mis tegeleb kõikide valdkondade standardimisega
QR-kood	*Quick response code*, masinloetav kahemõõtmeline maatrikskood, mis võimaldab skaneerida informatsiooni
SQL	*Structured query language*, programmeerimiskeel andmebaasiga suhtlemiseks
Vipps	Norra mobiilimakse rakendus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Toodete otsmiseks ja teenuste kasutamiseks ettenähtud mobiilirakenduste ehk äppide arv maailmas kasvab pidevalt. See pakub kasutajatele kiiret ja lihtsat viisi iseseisvalt toodete ja teenuste eest maksmiseks ja samuti pakkujale lihtsat ning automatiseeritud viisi kõikide tehingute salvestamiseks. See omakorda tähendab, et toodet või teenust pakkuval ettevõttel on võimalik manuaalse töö pealt töötajate arvu ja aega kokku hoida.

Käesoleva töö raames luuakse restoranide külastamise ajal mobiilsetel seadmetel kasutamiseks ettenähtud veebirakendus, mis täidaks olemasoleva mobiilirakenduse eesmärki, kuid võimaldaks seda teha kiiremalt ja mugavamalt ilma rakendust alla tõmbamata ja telefonisse paigaldamata. Rakendusega on võimalik broneerida laudu, sirvida menüüd, tellida menüüst tooteid, maksta oma tellimuse eest ja soovi korral jätta jootraha. Veebirakenduse soov tuleb kliendilt endalt, kes tõi põhjusena välja, et restoranide külastajad ei ole nõus mobiilirakendust alla tõmbama.

Rakenduse eesmärk on lihtsustada ja minimaliseerida teenindajate tööd ning suurendada rakenduse ja restoranide külastajate arvu pakkudes paremat kasutajakogemust. Iga külastaja, kes kasutaks rakendust, hoiaks kokku teenindajate aega menüüde jagamise, tellimuste küsimise ja kirjapanemise, arvete jagamise ning maksmisele kuluva aja pealt. Samuti paraneks ka kasutajate külastuskogemus pakkudes neile kiiremat ja mugavamat viisi teenuse kasutamiseks.

Kõigepealt antakse käesolevas töös ülevaade olemasolevast mobiilirakendusest. Seejärel kirjeldatakse töö raames arendatava veebirakenduse tausta, ülesande püstitust ja eesmärke. Seejärel kirjeldatakse rakenduse nõuded, ülesehitus ja arenduseks kasutatud programmeerimiskeelte, tehnoloogiate ning tööriistade valikuid. Seejärel antakse ülevaade rakenduse arendusest ja realisatsioonist.

2 Taust

Arendatav rakendus kuulub Norra ettevõttele. See sai alguse, kui samale ettevõttele kuuluv restoran soovis enda teenindajate tööd lihtsustada ja klientide külastuskogemust parandada. Selleks arendati algselt kolm rakendust:

- Mobiilirakendus, mis on mõeldud kasutamiseks restoranikülastajate poolt. Selle rakendusega on võimalik sirvida menüüd, valida välja sobivad toidud ja joogid, esitada tellimus, tellimuse eest maksta ning soovi korral jätta jootraha.

- Veebirakendus, mis on mõeldud kasutamiseks restoranis töötavate teenindajate poolt. Selles rakendus kuvatakse külastajate tellimused, et anda tellimus edasi kööki valmistamiseks. Tellimuse juures kuvatakse lauanumber, et valmis see kliendile lauda viia. Samuti oli võimalik läbi teenindajarakenduse sooritada sularaha makseid, kui klient ei soovinud seda läbi mobiilirakenduse teha.

- Veebirakendus, mis on mõeldud kasutamiseks restoranide administraatorite poolt. Selle rakendusega on võimalik lisada ja muuta informatsiooni restoranide kohta, näiteks menüüde sisu (toidud, joogid, hinnad, pildid, allergeenid), restoranide lahtiolekuajad ja muud.

Huvi rakenduse vastu kasvas oodatust suuremaks ning lõpuks kasvas sellest ühele restoranile mõeldud rakendustest välja eraldi platvorm, mida ka teised restoranid said kasutama hakata.

3 Probleem

Nagu eelnevalt mainitud, lihtsustab rakenduse kasutamine teenindajate tööd. Mida rohkem külastajaid kasutab rakendust, seda vähem on teenindajatel tööd vaja teha, mis omakorda tähendab seda, et teenindajaid on vähem vaja. Nii aitaks rakenduse kasutamine külastajate poolt restorani majanduslikult. Lisaks restoranile on rakendus kasulik ka külastajatele. Kõik toimingud kuni toidu lauda toomiseni on võimalik rakendusesiseselt külastajal endal ära teha. Seal hulgas broneerida laud, sirvida menüüd, sooritada tellimus ja selle eest ka tasuda. Mitte üheski nendest toimingutest ei ole vaja külastajal oodata teenindajat ja loota, et tellimus sai korrektselt kirja. Viimaks on kõik tellimused sellisel juhul salvestatud andmebaasi ja hiljem saadaval.

Probleem seisneb selles, et märkimisväärselt suur osa restoranide külastajaid ei ole nõus mobiilirakendust alla tõmbama. See tähendab seda, et kõik eelnevalt nimetatud kasutegurid võib maha tõmmata. Põhjuseid, miks külastajad ei ole nõus rakendust alla tõmbama, võib olla mitmeid. Mobiilirakenduse suurus antud hetkel on 97 MB. On mõistlik eeldada, et antud rakendus laetakse alla esimest korda restorani külastades. Nii suure rakenduse puhul võib allalaadimine olla ajaliselt pikk, kui selleks kasutada WiFit või mobiilset andmesidet. Samuti piiratud andmemahu puhul ei soovi kasutajad oma internetti selle peale kasutada. Samuti võib põhjuseks olla see, et restorane ei külastata piisavalt, seega puudub põhjus rakendust ühekordseks kasutamiseks alla tõmmata.

Lahendusi antud probleemide jaoks võib olla mitmeid, kuid klient pöördus meie poole sooviga arendada sarnase välimuse ja funktsionaalsusega mobiilseadmetel kasutamiseks ettenähtud veebirakendus. Veebirakenduse puhul puudub vajadus seda alla tõmmata ja enda telefoni paigaldada, puudub isegi vajadus antud lehekülje aadressi veebibrauserisse sisse trükkida, kuna rakenduse avamiseks saab kasutada ka QR-koodi.

4 Analüüs

Antud peatükis kirjeldatakse valmiva rakenduse nõudeid, vooskeemi ja olemi-suhte diagramme, analüüsitakse andmebaasi, ees- ja tagarakenduse tehnoloogiate ning kasutajaliidese raamistike valikuid.

4.1 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded lepiti kokku kliendiga enne rakenduse arenduse algust.

- Rakendus on kasutatav mobiilsetel seadmetel.
- Rakendust on lihtne kasutada ning selle voog oleks loogiline ja arusaadav.
- Rakendus on optimeeritud ja jõudluse poolest kiire.

4.2 Funktsionaalsed nõuded

Funktsionaalsed nõuded lepiti kokku kliendiga ja tiimi analüütikuga suuremas osas enne rakenduse arenduse. Nõuded lisati ja täiendati ka arenduse ajal.

- Veebirakenduse avalehel kuvada avatud restoranid eespool, suletud restoranid allpool ja see kasutajale eraldi selgelt välja tuua.
- Menüüsid peab olema võimalik kategooriatele põhjal filtreerida.
- Ostukorvi peab saama lisada sama toodet erinevate lisanditega.
- Sisselogimine ja maksmine peab toimuma kasutades Vippsi mobiilimaksete rakendust ning seda enne tellimuse esitamist.
- Kasutajal ei ole võimalik sisestada ebakorrektseid sisendeid, nt jootraha summa, mis on väiksem või võrdne nulliga.
- Probleemide ja vigade tekkimisel kuvada kasutajale selged veateated.

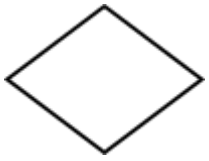
4.3 Rakenduse vooskeem

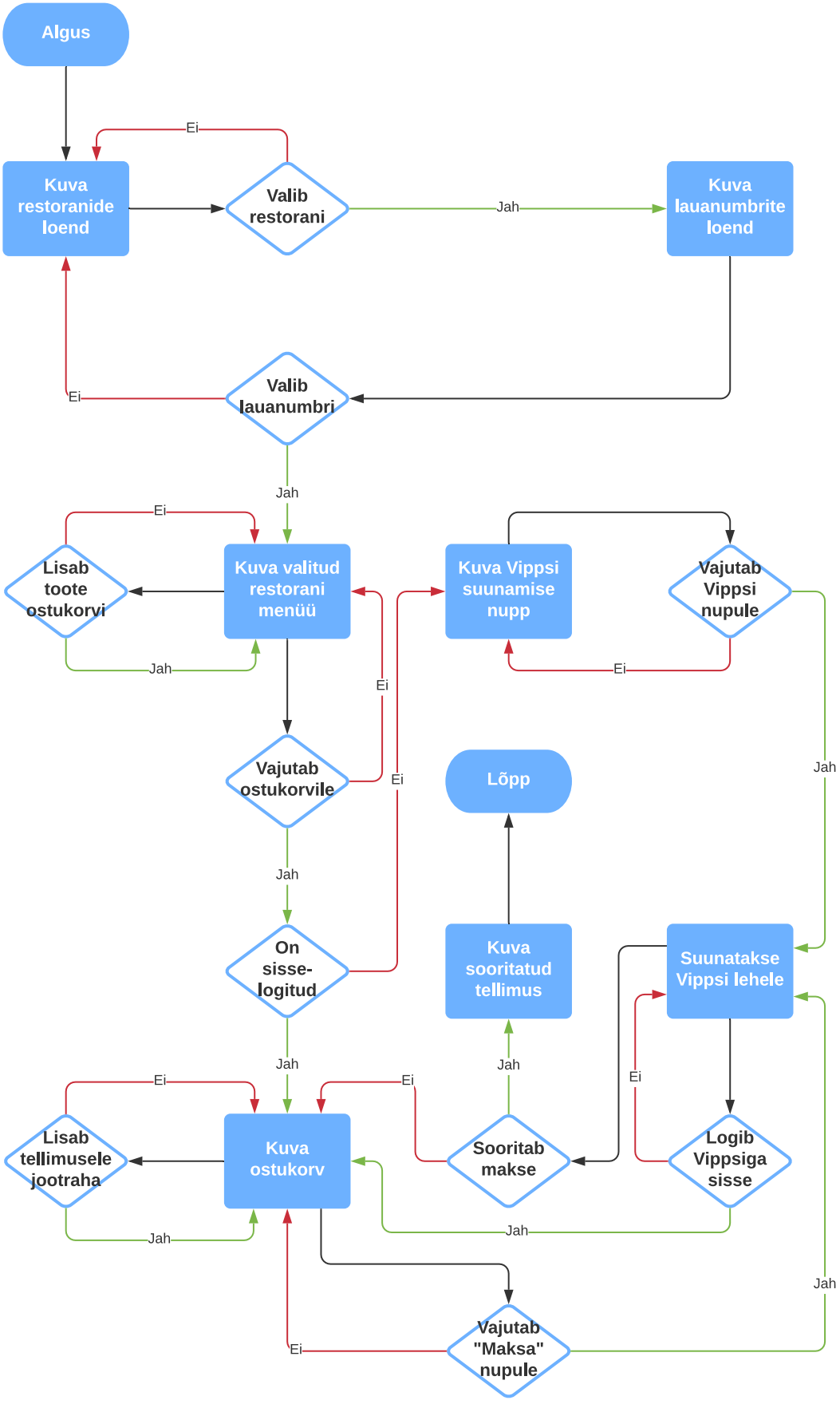
Enne töövahendite valikut ja rakenduse arenduse algust on oluline selgeks teha, milline peab olema rakenduse käitumine. Selle modelleerimiseks on kasutatud vooskeemi (*flowchart*).

Üldiseid vooskeeme on lihtne koostada ja need annavad hea visualiseeritud ülevaate rakenduse käitumisest. Vooskeeme koostatakse kasutades ISO (*International Organization for Standardization*) poolt määratud sümboleid ja kujundeid, mis omavad kindlat rolli või kirjeldavad kindlat olukorda skeemis [21].

Joonisel 2 oleva vooskeemi loomiseks kasutavate kujundite kirjeldus on tabelis 1.

Tabel 1. Kasutatud standardiseeritud vooskeemi elemendid.

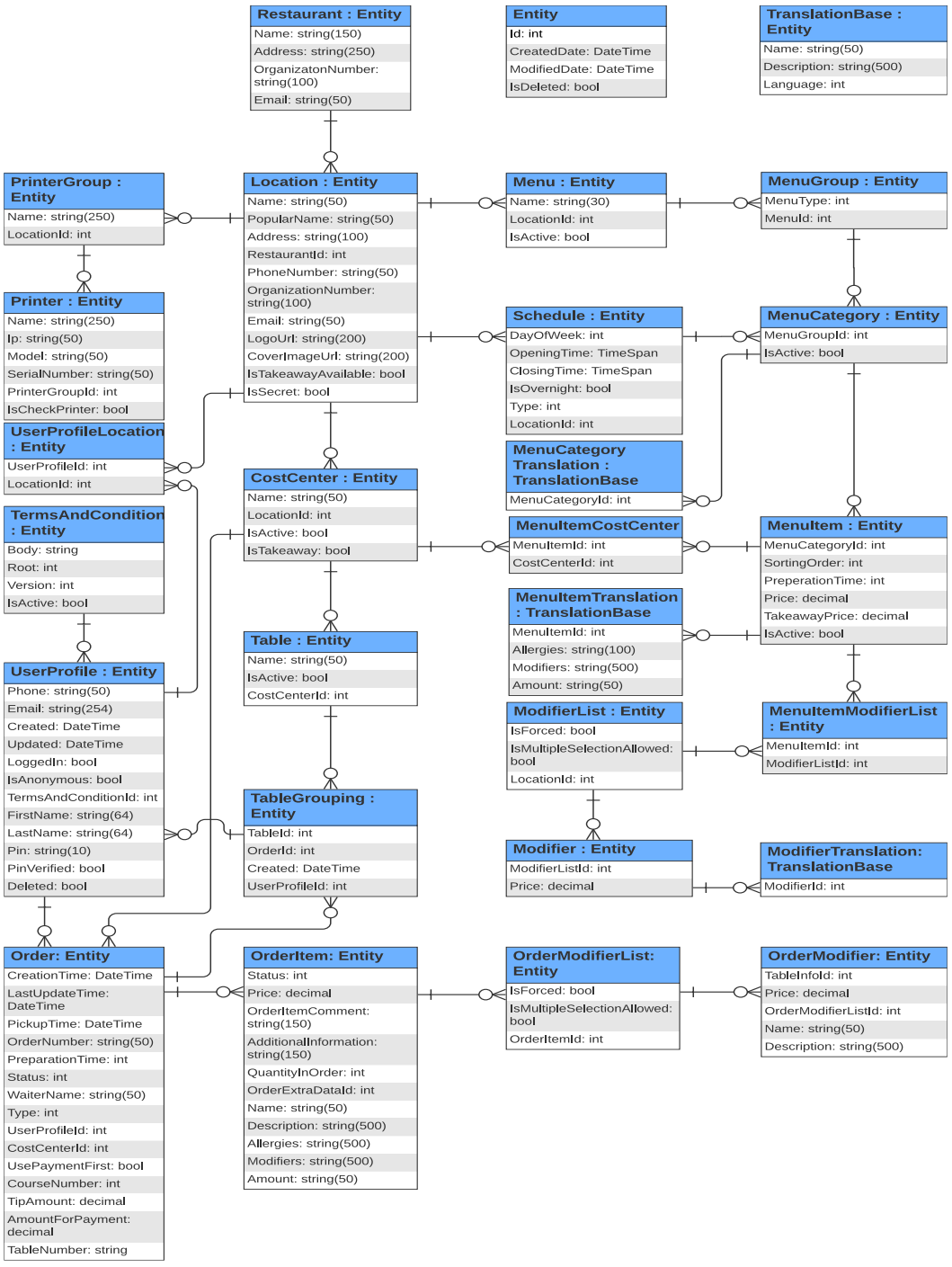
ISO kujund	Nimi	Kirjeldus
	Otspunk	Tähistab rakenduse algus- või lõpp-punkti.
	Voosuund	Näitab, millises suunas rakenduse käitumine toimub.
	Tegevus	Kirjeldab tegevust, mida rakendus teeb.
	Otsus	Kirjeldab valikut, millest oleneb rakenduse järgnev käitumine.



Joonis 1. Rakenduse vooskeem. (Allikas: autori koostatud)

4.4 Rakenduse olemi-suhte diagramm

Kuigi autori ja arendustiimi kasutuses on andmebaas, mida kasutab olemasolev mobiilirakendus, on hea idee enne rakenduse arendust selgeks teha, kuidas andmebaasi struktuur välja näeb. Selleks on kasutatud ERD (*entity relationship diagram*) ehk olemi-suhte diagrammi metoodikat. Joonisel 3 on kujutatud kasutatava andmebaasi andmeloogiline ERD mudel. Mudel on loodud kasutades rakendust Lucidchart.



Joonis 2. Rakenduse andmebaasi olemi-suhte diagramm. (Allikas: autori koostatud)

4.5 Andmebaasi valik

Andmebaasi valikuks on PostgreSQL. Otsus on siin lihtne, kuna on võimalus kasutada olemasoleva rakenduse andmebaasi koos test- ja pärisandmetega. See hoiab kokku suure hulga tööd projekti algfaasis. Samuti tuleb meeles pidada, et arendatav veebirakendus tuleb lisaks, mitte asenduseks, mobiilirakendusele, seega tahame me kõiki rakendusega seotud andmeid hoida ühes kohas. Kahe andmebaasi olemasolul oleks andmete muutmine ja lisamine ajakulukam ning oleks oht vigade tekkimiseks.

PostgreSQL on tasuta avatud lähtekoodiga objekt-relatsiooniline andmebaasisüsteem, mille juured said alguse 1986. aasta „POSTGRE“ projektist California ülikoolis ning seda on praeguseks arendatud rohkem kui 30 aastat. PostgreSQL on tuntud oma arhitektuuri, töökindluse ja andmete tervikluse poolest [9].

Relatsiooniline andmebaas on andmetekogum koos eelnevalt määratletud seostega. Andmeid hoitakse tabelites, mis koosnevad ridadest ja veergudest. Iga veerg tabelis kirjeldab andmevälja ning iga rida tabelis on kirje, kus kirje on struktureeritud info kindla objekti kohta [12] [13].

4.6 Tagarakenduse programmeerimiskeele ja tehnoloogia valik

Tagarakenduse tehnoloogiate valikul analüüsitakse täpsemalt kahte kõige populaarsemat tehnoloogiat, millega töö autor on eelnevalt koolis ja iseseisvalt kokku puutunud.

Java on objektorienteeritud ja staatiliselt tüübitud programmeerimiskeel, mis loodi Sun Microsystemsi (hiljem omandatud Oracle poolt) poolt 1991. aastal, kui James Gosling üritas leida viisi arendada C++ programme, mis jookseks võimalikult suure hulga platvormide peal ilma koodi uuesti kompileerimata [14].

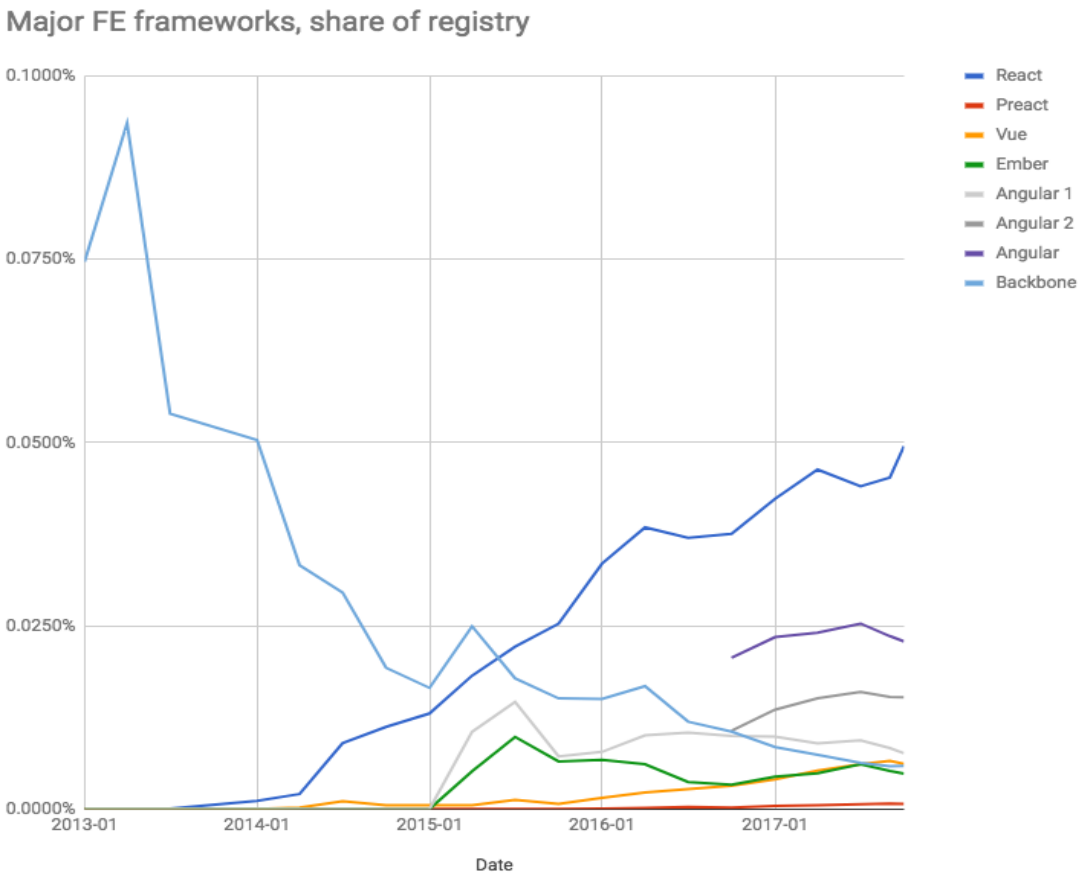
C# on objektorienteeritud ja staatiliselt tüübitud programmeerimiskeel, mis loodi Microsofti poolt 2000. aastal. Veebirakenduste arendamiseks saab C# programmeerimiskeeles kasutata .NET raamistikku. .NET on moderne, innovaatiline ja avatud lähtekoodiga arendusplatvorm, mis võimaldab arendada kõrgekvaliteedilisi rakendusi kiirelt ning on 2019. ja 2020. aasta Stack Overflow arendajate küsitluste järgi valitud enim meeldivamaks raamistikuks [15].

Tagarakenduse puhul kindlad lähtetingimused puuduvad, kuid sarnaselt andmebaasi valikule on tiimil võimalus kasutada juba olemasoleva tagarakenduse koodi, mis on kirjutatud C# programmeerimiskeeles kasutades .NET raamistikku. See annab võimaluse taaskasutada juba olemasolevat loogikat, kuna arendatav veebirakendus peab oma funktsionaalsuselt olema sarnane mobiilirakenduse omale. Mõnes olukorras võib see olla ka negatiive, kuna juurdearendust tehes peab olema kindel, et mobiilrakenduse funktsionaalsust katki ei tehtaks. Samuti on arendustiimis .NETi raamistiku kogemusega arendajad ja ka töö autor tunneb ennast kõige kindlamalt .NETi raamistiku peal.

4.7 Eesrakenduse programmeerimisekeele ja tehnoloogia valik

Veebiarenduse kontekstis peetakse eesrakenduse all silmas veebilehe graafilist kasutajaliidest, läbi mille on kasutajal võimalik veebilehega suhelda. Eesrakendus kuvab tagarakendusest (serverist) päritud informatsiooni ning kuvab selle kasutajale nähtavaks. Samuti saab kasutaja läbi eesrakenduse saata veel erinevaid päringuid, et andmeid lisada, muuta või kustutada [8].

Võrreldes tagarakenduse programmeerimiskeelte valikuga on eesrakenduse puhul see väiksem. Populaarseimaks programmeerimiskeele valikuks on JavaScript. Kuigi keelte valik on väiksem, siis on valik kordades suurem raamistike hulgas. Selle töö raames keskendutakse kolme eesrakenduse raamistiku analüüsile: React, Angular ja Vue [7].



Joonis 3. Eesrakenduse raamistike populaarsus. [7]

Reacti, Angulari ja Vue puhul on tegemist hetkel kõige enim kasutatavate JavaScirpti raamistikega. Kindla raamistiku populaarsus üksi ei ole kindlasti põhjus, miks seda valida, kuid raamistiku populaarsus tähendab suuremat kommuuni ja suuremat hulka arendajaid, kes jagavad informatsiooni, lahendusi ja vastuseid tekkivate probleemide ja küsimustega [7].

Reacti puhul on tegemist kasutajaliideste ehitamiseks mõeldud teegiga, mis on loodud Facebooki poolt 2013. aastal. Just nimelt teegiga, mitte raamistikuga. Erinevust raamistiku ja teegi vahel ei ole universaalselt määratud, kuid enamasti peetakse selle all silmas kahte tegurit - *Inversion of Control* ja arhitektuurilised otsused. On olemas ka Reactil põhinevad raamistikud nagu näiteks Next.js [1]. React kirjeldab ennast enda koduleheküljel raamistikuna, mis on deklaratiivne, komponendipõhine ja järgib “Learn Once, Write Anywhere” printsiipi [2].

Angular (ka Angular 2+ või Angular CLI) on TypeScriptil põhinev komponendipõhine ühelehe veebirakenduste loomiseks mõeldud raamistik, mis on arendatud Google poolt

2016. aastal [3]. Angular sobib nii väiksemate kui ka suuremate rakenduste arendamiseks ning see on loodud eesmärgiga, et juurdearendust saaks teha võimalikult vähese vaevaga. Angulari arendajaid on kokku üle 1,7 miljoni, millest võib järeldada, et tekkivate probleemide ja küsimuste puhul on lihtne ja kiire leida vastuseid [2].

Vue on kasutajaliideste ja ühelehe veebirakenduste loomiseks mõeldud JavaScripti raamistik [4]. Vue loojaks on Evan You, kes töötas enne Googles arendajana ning kasutas mitmetel projektidel töötamisel AngularJSi raamistikku. Tema idee sai alguse, kui soovis võtta välja kõik head kontseptsioonid AngularJSist, et ehitada midagi väga lihtsat [5]. Vue kirjeldab ennast enda koduleheküljel raamistikuna, millega on lihtne alustada, mis on paindlik ja kiire jõudlusega [6].

Eesrakenduse puhul ühtegi lähtetingimust ette ei ole antud ja selle puhul on valik täielikult arendustiimi enda teha. Arvestades antud töö jooksul valmiva rakenduse skoopi ja Reacti raamistiku populaarsust, tänu millele on sagedaste probleemidele lahenduse leidmine kiire ning nii töö autori kui ka tiimiliikmete eelnevat kogemust Reacti raamistikuga, on tegemist parima valikuga. Vue või Angulari kasutamine nõuaks õppimist ja aega, mis pikendaks rakenduse arenduse kulgu. Samuti on uut raamistiku kasutades oht, et ei järgita parimaid tavasid. See tähendab, et rakenduse kasvades võib uute funktsionaalsuste implementeerimine olla keeruline ja tekib palju nii öelda tehnilist võlga (*technical debt*).

4.8 Kasutajaliidese raamistiku valik

Kuna valmiv veebirakendus peab kindlasti olema kasutatav mobiilsetel seadmetel, siis on kasutajaliidese raamistiku valik väga tähtis. Valiku tegemisel arvestatakse sellega, et antud raamistikuga saaks koodiridade hulga hoida võimalikult väiksena. Mida vähem koodiridu, seda puhtam, loetavam ja arusaadavam kogu kood on. Samuti peaks raamistik olema võimalikult konfigureeritav vastavalt vajadusele, kuna arendatav rakendus kuulub kindlale ettevõttele, peab olema seda vastavalt brändile võimalikult lihtne muuta.

Loodava rakenduse kasutajaliidese raamistikuks kasutatakse Material-UI, kuna see täidab kõige paremini eelnevalt kirja pandud eesmärke. Komponentide disaini määramine on lihtne ja see hoiab koodiridade hulga võimalikult väiksena. Samuti on Material-UI teegi poolt pakutavad komponendid piisavad rakenduse kasutajaliidese ehitamiseks.

5 Arendus

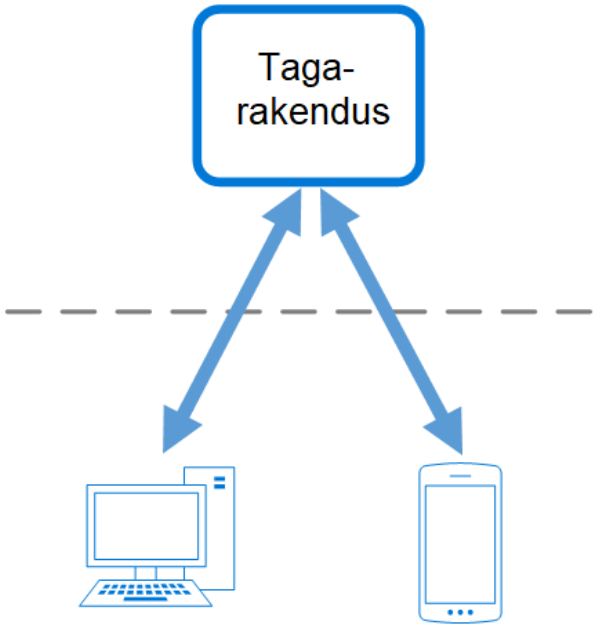
Antud peatükis antakse ülevaade kasutatavast disainimustrist, ees- ja tagarakenduse arendusest ja rakenduse testimisest.

5.1 Backends for Frontends muster

Backends for Frontends on disainimuster, mille eesmärk on pakkuda parimat kasutajakogemust. Seda eesmärki täidetakse luues tagarakendus, mis serveerib andmeid ainult ühele kindlale liidesele. Nii saab piirata päringute hulka ja suurust. Tagarakendus saadab täpselt nii vähe andmeid kui võimalik ja nii palju kui vajalik. Tulemus on kiirem rakendus ja väiksem ning lihtsamini hallatav koodibaas [17].

5.1.1 Probleem

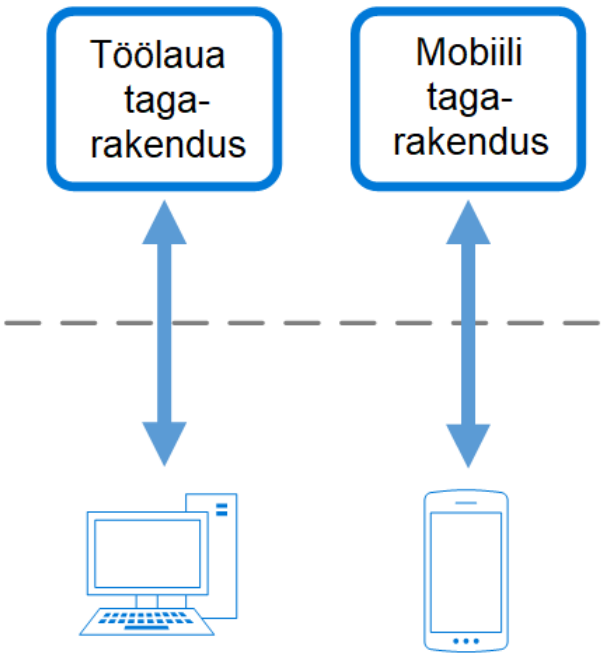
Rakendus võib olla algselt arendatud ühele kindlale platvormile, näiteks töölauale. Kasutajate arvu kasvu tõttu otsustatakse teha ka mobiilirakendus, mis kasutab sama tagarakendust. Nüüd muutub tagarakendus üldotstarbeliseks, serveerides mitut erinevat klienti. Aga erinevatel klientidel võivad vastavalt platvormile olla erinevad piirangud. Selline lahendus on pikemas perspektiivis halb, kuna nendest piirangutest tulenevalt on tagarakenduse jaoks erinevad nõuded. Juurdearendused muutuvad seetõttu raskemaks ja ajakulukamaks, kuna uute muudatuste lisamisel on vaja kindel olla, et ühe või teise liidese funktsionaalsust katki ei teha [16].



Joonis 4. Jagatud tagarakendus. Autori tõlge allikast [16].

5.1.2 Lahendus

Lahendus on luua üks tagarakendus ühe klientrakenduse jaoks. Nii on võimalik tagarakendust muuta ja optimeerida vastavalt vajadusele. Selle tulemusena on rakendus kiirem, väiksem ja lihtsamini hallatav. Samuti on rakenduse kallal töötaval tiimil suurem vabadus ja kontroll arenduse üle [16].



Joonis 5. Backends for Frontends muster. Autori tõlge allikast [16].

5.1.3 Disainimustri kasutamine

Selle mustri kasutamise peale tuleks mõelda kui:

- tagarakendust kasutavad mitmed erinevad klientrakendused
- on vajadus optimeerida tagarakendust kindlate nõuete põhjal
- on vajadus teha tagarakendusele muudatusi, et erinevaid klientrakendusi toetada

See muster ei ole sobiv kui:

- erinevad klientrakendused teevad samu päringuid
- tagarakendust kasutab ainult üks klientrakendus [16]

Antud töö raames on Backends for Frontends mustri kasutamine õigustatud. Kuigi veebirakendus arendatakse samuti mobiilsetel seadmetel kasutamiseks, ei ole arendustiimil ega kliendil mingit soovi olemasoleva mobiilirakenduse funktsionaalsust muuta. Samuti oli kliendi üks soovidest, et arendatav rakendus oleks paremini optimeeritud ja kiirem. Uue API loomisel saame ees- ja tagarakenduse vahel toimuvate päringute suurused hoida nii väiksed kui võimalik.

5.2 Tagarakenduse arendus

Tagarakendus on kirjutatud kasutades .NET 5 raamistikku ja C# programmeerimiskeelt. Integreeritud arenduskeskkonnana kasutati JetBrainsi Riderit.

5.2.1 Struktuur

Tagarakendus on ülesehitatud mitmekihilisel arhitektuuril. Eraldatud on veebipõhine API, äriloogika ja andmed. Selline arhitektuur võimaldab koodibaasi hoida hallatavana ja uue funktsionaalsuse lisamise lihtsana.

- WebAPI – sisaldab veebipõhiseid API kontrollereid, läbi mille eesrakendus tagarakendusega suhtleb. See on arendatud kasutades BFF disainimustrit, st et iga lõpp-punkt tagastab täpselt nii palju andmeid kui eesrakendus kuvab.

- Services – äriloogikakiht, mis seob oma vahel andmekihi ja veebipõhised API kontrollerid. Sisaldab meetodeid, mida kasutatakse iga päringu puhul.
- Infrastructure – andmete juurdepääsu kiht. Sisaldab meetodeid, millega on võimalik andmebaasist andmeid küsida ja salvestada.
- Data Access – sisaldab domeeni olemeid ja andmebaasimigratsioone.
- Integration – sisaldab suhtlust printerite ja Vippsi API lõpp-punktidega.
- Unit Tests – sisaldab üksusteste.

5.2.2 Vippsi integratsioon

Kliendi soovil toimub kasutajate identifitseerimine ja tellimuste eest maksmine läbi Vippsi rakenduse. Sularaha- ja kaardimaksed antud töö skoopi ei kuulu.

Vipps on Norra mobiilimakse rakendus, mis tuli välja 30. mail 2015. Vipps võimaldab kasutajatel teha ülekandeid kasutades saaja pangakonto asemel telefoninumbrit. Praeguse seisuga on see suurim maksete rakendus Norras [19]. 2019. aasta septembri seisuga kasutab Vippsi 78% Norra elanikest [20].

Vippsi integreerimine tehti kasutades Vippsi ametlikku dokumentatsiooni. Olemasolevast viiest APIst kasutati antud töö raames kahte – Vipps Login API ja Vipps Order Management API.

5.3 Eesrakenduse arendus

Eesrakendus on kirjutatud kasutades ReactJs teeki ja TypeScript programmeerimiskeelt. Integreeritud arenduskeskkonnana kasutati VisualStudio Code-i.

5.3.1 Struktuur

Failide struktureerimise eesmärk hoida koodibaas hallatavana ja loetavana, et uute vaadete ja funktsionaalsuste lisamine oleks lihtne ning toimuks ilma erilise vaevata. Kuigi Reacti dokumentatsioonis on kirjas, et antud teegi kasutamisel puudub kindel failide struktureerimise viis, on seal siiski välja toodud populaarsemad valikud kuidas seda võiks teha [18].

Grupeerimine vaadete järgi. Selliselt struktureerides hoitakse ühes kaustas kõiki kindla vaatega seotud faile, sealhulgas stiilifailid, testid ja suhtlus tagarakendusega. Uue vaate lisamiseks tuleb lihtsalt luua uus kaust [18].

```
common/
  Avatar.js
  Avatar.css
  APIUtils.js
  APIUtils.test.js
feed/
  index.js
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  FeedAPI.js
profile/
  index.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
  ProfileAPI.js
```

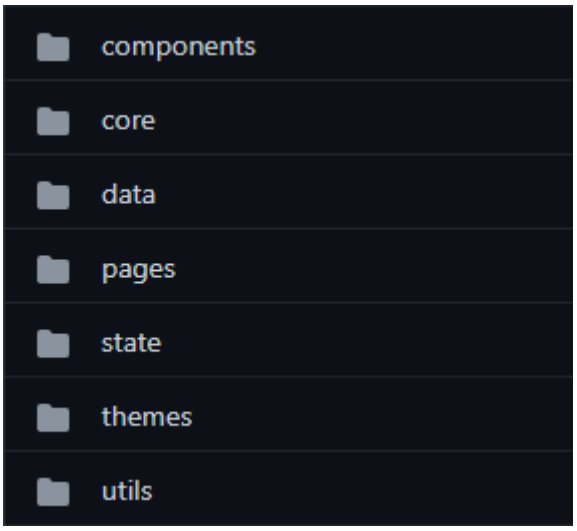
Joonis 6. Struktureerimine vaadete järgi [18].

Grupeerimine failitüüpide järgi. Selle asemel, et jagada vaated eraldi, jagatakse antud näites eraldi tagarakendusega suhtlus ja komponendid. Nii aga muutub komponentide kaust iga uue vaate lisamisel suuremaks ning tulevikus võib uute vaadete lisamine või muudatuste tegemine olemasolevatele vaadetele muutuda keerukaks ja ajamahukaks [18].

```
api/
  APIUtils.js
  APIUtils.test.js
  ProfileAPI.js
  UserAPI.js
components/
  Avatar.js
  Avatar.css
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
```

Joonis 7. Struktureerimine failitüüpide järgi [18].

Antud töö raames loodava rakenduse failide struktureerimiseks kasutatakse kooslust kahest eelnimetatud viisist. Kõigepealt grupeeritakse failitüüpide järgi, see järel vaadete järgi.



Joonis 8. Eesrakenduse failide struktuur.

- Components – sisaldab Reacti komponente, mida on võimalik üle rakenduse kasutada. Komponendid võivad olla üldotstarbelised, näiteks lehekülje päis ja navigatsiooniriba või ühe vaate spetsiifilised, näiteks toote lisanditeloend.

- Core – sisaldab üle rakenduse vajaminevaid liideseid, andmemudelid ja navigatsioonimeetodeid.
- Data – sisaldab andmeedastusobjekte ja teenuseid tagarakendusega suhtlemiseks. Selleks kasutatakse React Query teeki, mis lihtsustab päringute tegemist ja andmete *cache*-mist.
- Pages – sisaldab Reacti komponente, mis hoiavad endas ühte kindlat vaadet, näiteks menüüvaade.
- State – sisaldab rakenduse olekuga seotud faile ja Reacti kohandatud *hook*e.
- Themes – sisaldab Material-UI komponenditeegi disaini muutmiseks mõeldud konfiguratsiooni.
- Utils – sisaldab ülerakenduse vajaminevaid meetodeid nagu näiteks arvutused hindade ja kuupäevadega.

5.3.2 Kasutatavus mobiilsetel seadmetel

Antud rakenduse kasutatavus mobiilsetel seadmetel on väga olulise tähtsusega, kuna mobiilsed seadmed moodustavad väga suure osa kasutajatest. See tähendab, et rakenduse kujunduse loomisel tuleb vältida mobiilsetel seadmetel ebamugavust tekitavate elementide kasutamist nagu näiteks sisendväljad ja rippmenüüd. Pigem peaks need asendama elementidega, mida on võimalik vajutada või tõmmata.

5.4 Testimine

Testimine ja testide kirjutamine toimus rakenduse arenduse ajal jooksvalt. Arendustiimis kokkulepitult oli iga ülesande osa kirjutada lisatud funktsionaalsusele testid. Samuti testiti rakendust pärast selle valmimist kliendi poolt. Testimise eesmärk oli leida rakenduses esinevaid vigu.

5.4.1 Eesrakenduse testimine

Eesrakenduse puhul testiti olekuga seotud meetodeid, utiliite ja komponente kasutades Jest raamistikku.

Antud rakenduse puhul oli terve oleks seotud ostukorviga, seega oleku puhul testiti toodete ostukorvi lisamist ja eemaldamist, toote koguse ja lisandite muutmist.

Utiliitide puhul testiti ülerakenduse vajaminevaid meetodeid nagu näiteks hindade ja kuupäevade arvutused.

Komponentide testimisel kontrolliti, et vastavalt etteantud andmete põhjal käituks komponent vastavalt soovitult. Näiteks kontrolliti kui anda komponendile ette ostukorvis olevad tooted, siis toodete hindade summa oleks korrektne.

5.4.2 Tagarakenduse testimine

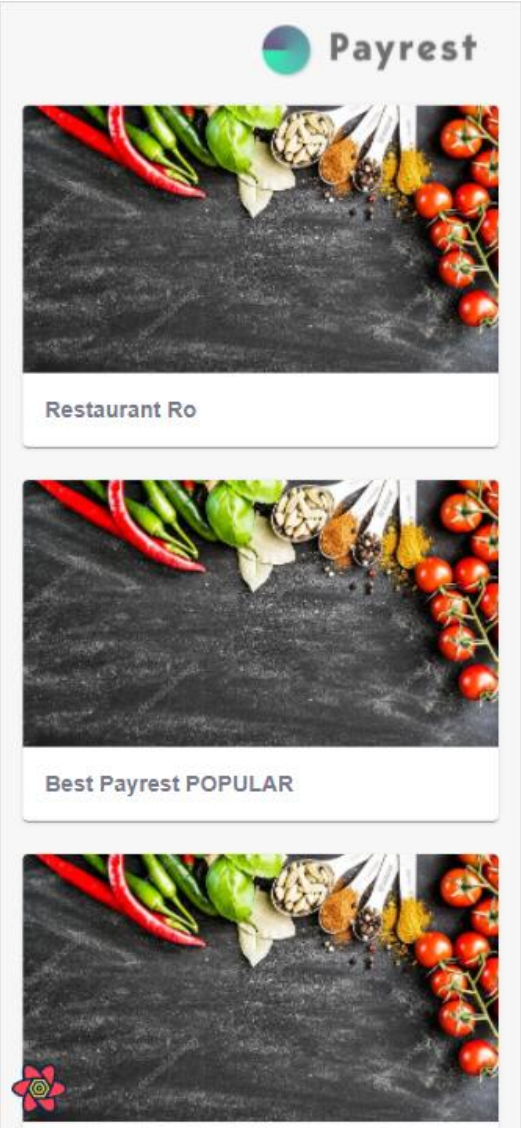
Tagarakenduse puhul testiti äriloogikakihi meetodeid kasutades NUnit raamistikku. NUnit on üksustestimisraamistik, mis on mõeldud kasutamiseks kõikide .NET programmeerimiskeeltega (C#, F#, Visual Basic) [22] [23].

6 Realisatsioon

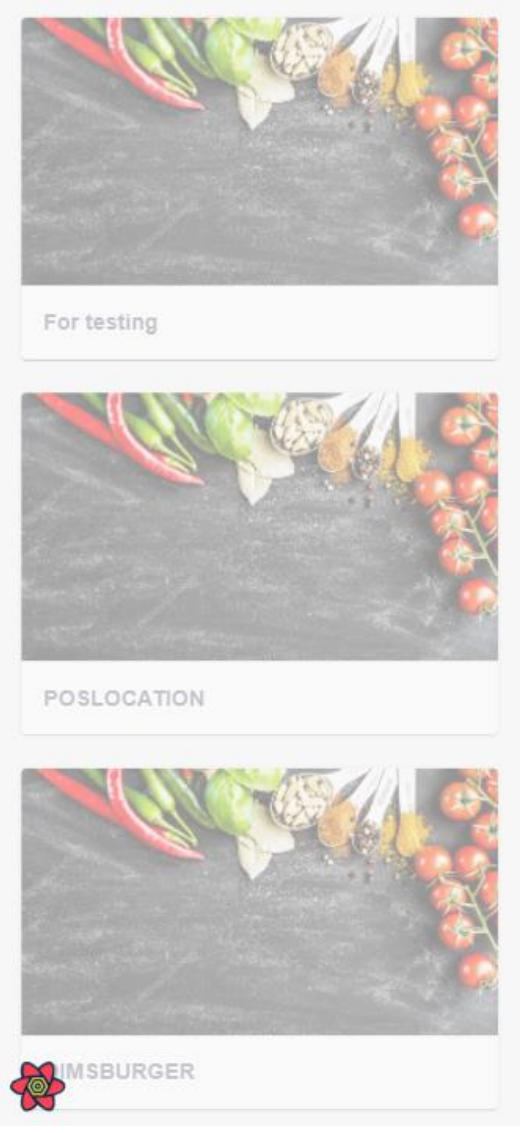
Antud peatükis antakse ülevaade rakenduse vaadetest, disainivalikutest ja funktsionaalsustest. Järgnevad näited põhinevad testandmetel.

6.1 Avaleht

Veebirakenduse avades kuvatakse kasutajale loend restoranidest. Loendi alguses kuvatakse lahtiolevad restoranid ja loendi lõpus kinniolevad restoranid tumedamalt.



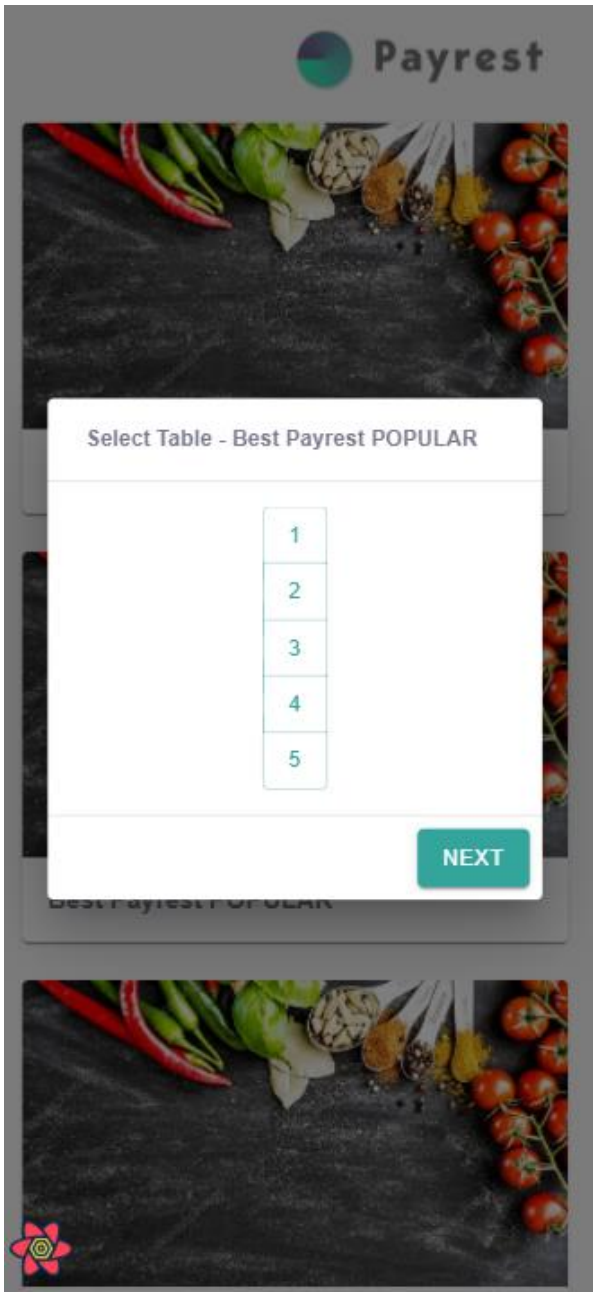
Joonis 9. Veebirakenduse avaleht.



Joonis 10. Hetkel suletud restoranide kuvamine.

6.2 Laua valimine

Restorani valimisel kuvatakse kasutajale loend vabadest laudadest. Kasutajal on võimalus valida lauanumber ning vajutada nupule „NEXT“ või sulgeda avanenud hüpikaken vajutades hüpikaknast välja.



Joonis 9. Vabade lauanumbrite loend.

6.3 Menüü

Laua valimisel kuvatakse kasutajale menüü vaade. Korraga kuvatakse terve menüü, mis on jagatud kategooriate järgi osadeks. Kasutajal on võimalik hüpata kindla kategooria osani vajutades sellele loendis, mis on kuvatud restorani informatsiooni all. Kui kategooriateloend sisaldab rohkem elemente kui korraga ekraanile mahub, siis on võimalik seda tõmmata horisontaalselt.

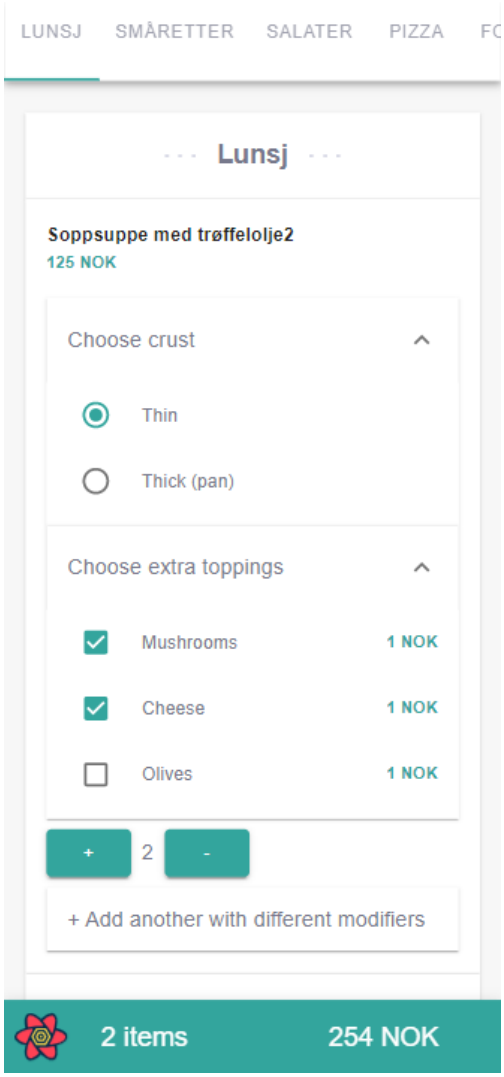
Enne kui kasutajal lastakse edasi minna ostukorvi vaatesse, peab olema ta sisse logitud.

Joonis 12. Menüü vaade sisse logimata.

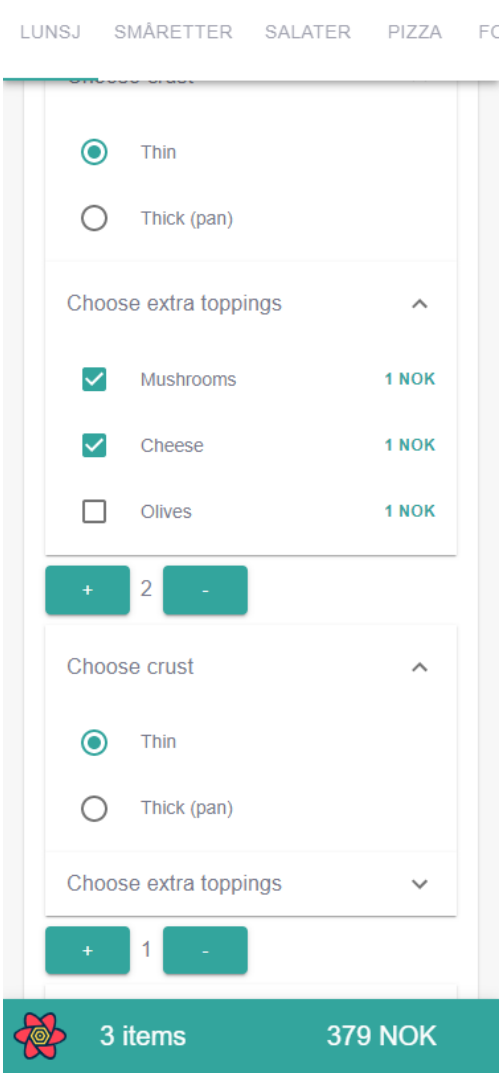
Joonis 13. Menüü vaade sisselogituna.

6.4 Toodete lisamine

Toote peale vajutamisel kuvatakse tootekirjeldus ja lisandid. Lisandigruppidel, mis on kohustuslikud (nt pitsapõhja paksus, liha küpsusaste), valitakse vaikimisi esimene lisand. Lisandigruppides, kus saab teha korraga mitu valikut, kuvatakse nende ees märkeruut. Kui lisandgrupis on lubatud ainult ühe lisandi valimine, siis kuvatakse selle ees raadionupp. Toote ostukorvi lisamisel kuvatakse „ADD“ nupu asemel antud toote kogus ostukorvis. Seda kogust on võimalik suurendada või vähendada. Kui kasutaja soovib lisada ostukorvi sama toodet, kuid erinevate lisanditega, siis selleks on nupp „Add another with different modifiers“.



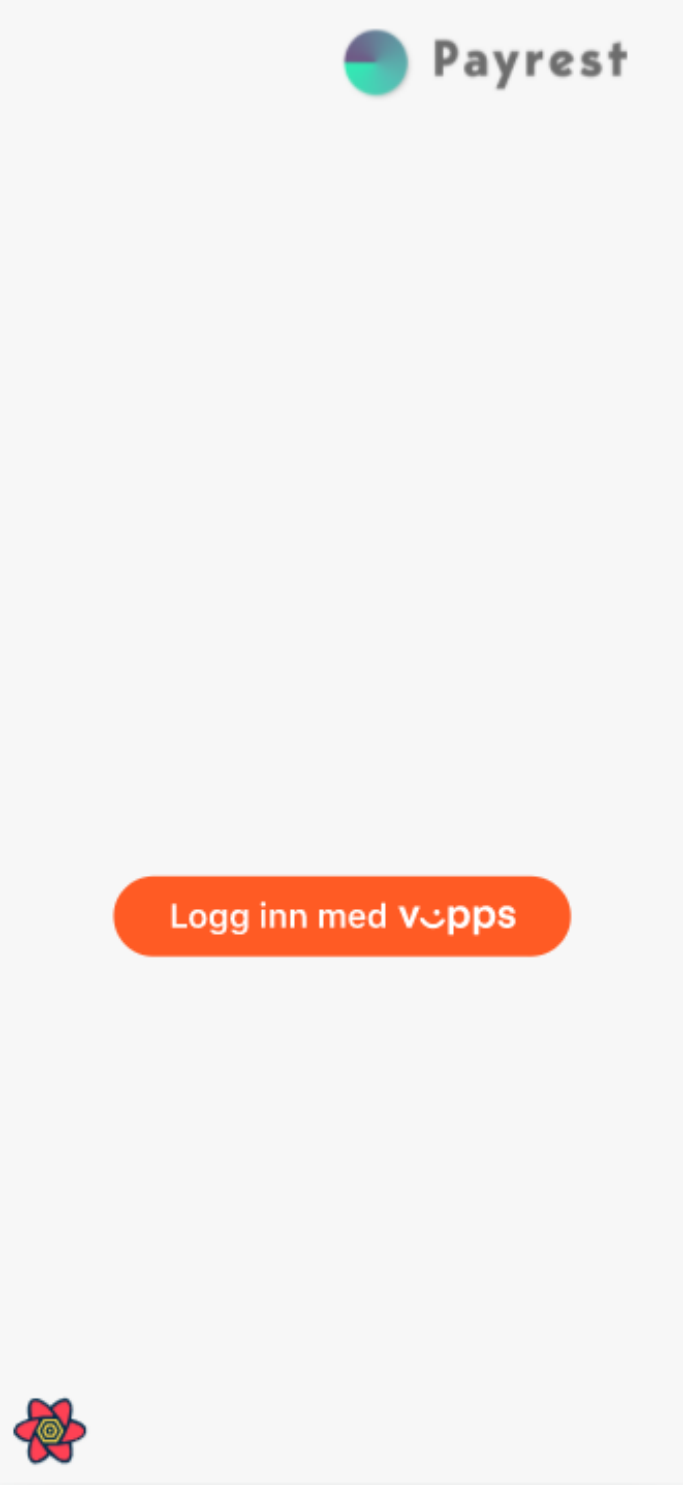
Joonis 14. Toote ostukorvi lisamine.



Joonis 15. Sama toote teise lisanditega ostukorvi lisamine.

6.5 Vippsiga sisselogimine

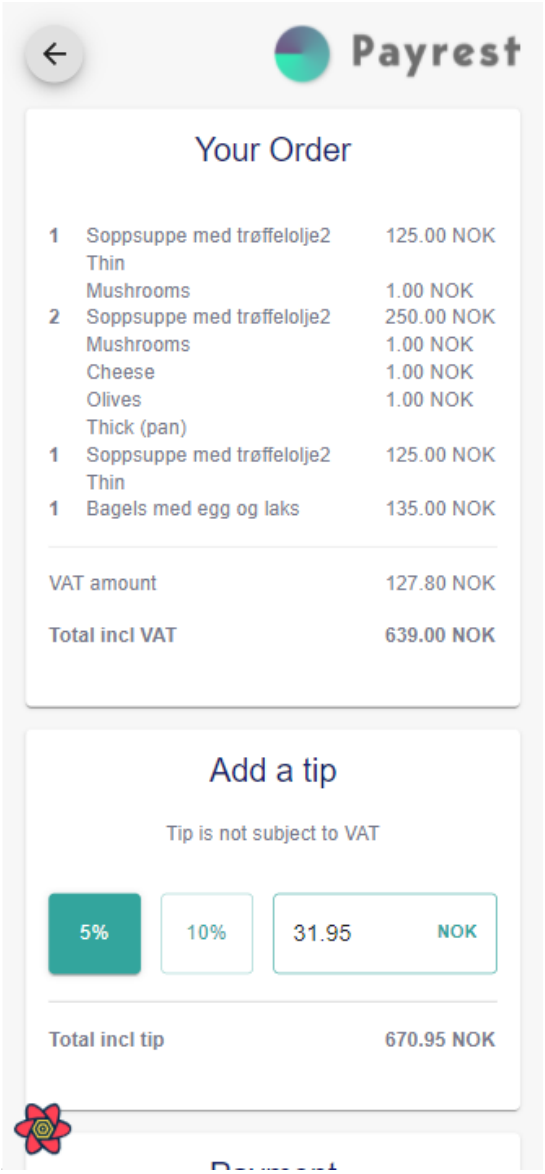
Ostukorvi vaateni ei ole võimalik enne jõuda, kuni kasutaja on sisse loginud. Sisselogimiseks kuvatakse kasutajale nupp, mis suunab ta Vippsi rakendusse.



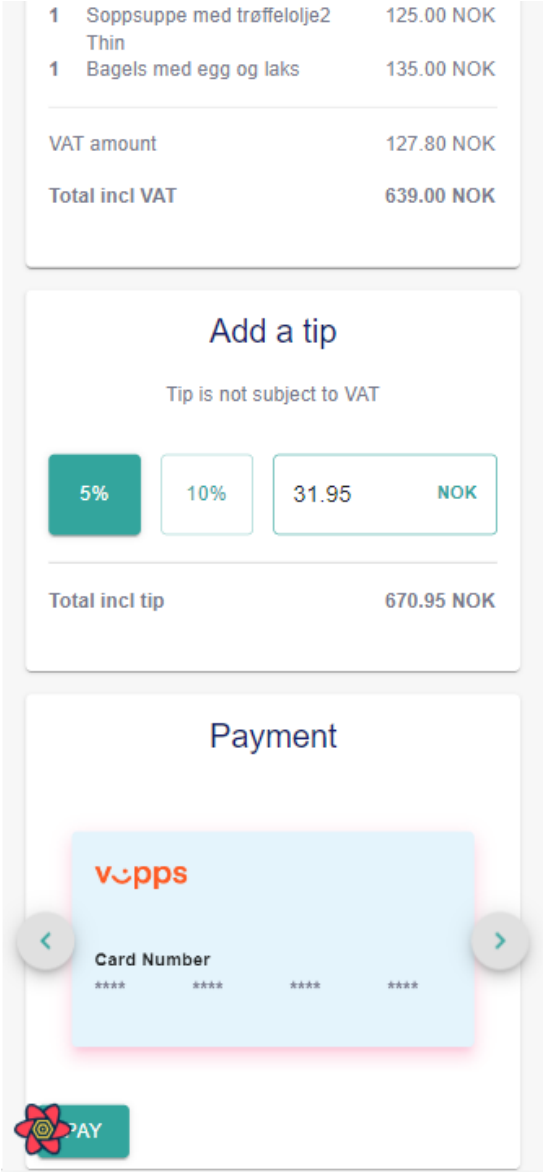
Joonis 16. Vippsiga sisselogimise nupp.

6.6 Ostukorv

Ostukorvi vaates kuvatakse kasutajale valitud lisatud tooted koos lisanditega ja hindadega. Toodete sektsioonist allpool on võimalik kasutajal soovi korral lisada jootraha. Mobiilsetel seadmetel kasutatavust silmas pidades on kasutajal võimalik jootraha lisada nupuvajutustega, vastavalt 5 ja 10 protsenti, kuid kui klient soovib, siis saab ta jootraha koguse ka ise valida. Jootraha sektsioonist allpool on kasutajal võimalik valida makseviis, kuid antud töö raames on selleks ainult Vippsi mobiilimakse.



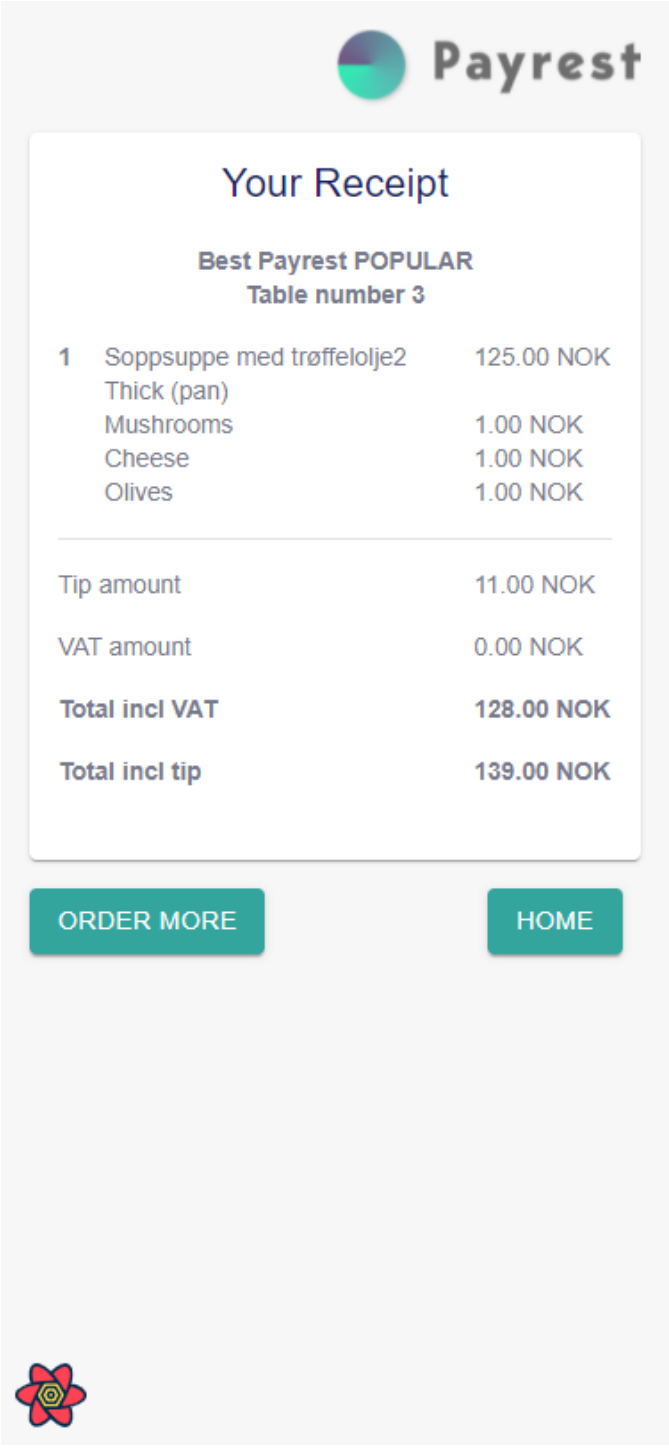
Joonis 17. Ostukorvi vaade 1.



Joonis 18. Ostukorvi vaade 2.

6.7 Tellimus

Tellimuse vaade kuvatakse kasutajale pärast edukat makset. Tellimuses olev info on nii öelda kuvatõmmis tellimuse hetkest. Antud toodete andmed salvestatakse eraldi tabelisse ja neid enam ei muudeta. Sellega hoitakse ära olukord, kus tulevikus võib toote hind muutuda ja tellimuse hind ei klapi enam makstud rahasummaga.



Joonis 19. Tellimuse vaade.

6.8 Rakenduse kasutegur

Kuna antud töö raames ei uuritud, kui palju aega kulus eelnevalt külastajatel tellimuste sooritamiseks, siis on raske täpsete numbritega välja tuua palju aega rakendus ja selle kasutamine säästab. See suuresti oleneb restoranist, hetkesest külastajate ning teenindajate arvust ja muudest sarnastest teguritest.

Võrreldes olemasoleva mobiilirakendusega säästab arendatud veebirakendus aega selle pealt, et puudub vajadus Play Store-st või App Store-st seda otsida, alla laadida ja telefoni paigaldada. Olenevalt internetiühenduse kiirusest hoitakse selle pealt juba aega kokku. Samuti on võimalik veebirakendus avada skaneerides restorani laudadel olevaid QR-koode, mis valivad kasutajale õige restorani ja lauanumbri automaatselt ära. Samuti disainiti veebirakendus lähtudes vana mobiilirakenduse välimusest ja funktsionaalsusest, seega puudub kasutajatel vajadus rakendust uuesti õppida.

Võrreldes teenindajate poolt tehtava tööga hoiab arendatud rakendus samuti aega kokku. Kõigepealt on võimalik külastajal teha peaaegu kõik teenindaja tööülesanded ise ära, näiteks menüüde jagamine, tellimuste esitamine ja maksmine, ning seda kiiremini ja mugavamalt. Samuti puudub tegevustevaheline ooteaeg. See mängib eriti rolli, kui restorani külastajate arv on suur.

Kokkuvõttes võib öelda, et rakenduse kasutamine on restoranide jaoks majanduslikult kasulik, vähendades nende jaoks vajaminevat töötajate hulka ja pakkudes samal ajal klientidele paremat külastuskogemust kiirema ja mugavama teenuse näol.

7 Kokkuvõte

Käesoleva töö eesmärk oli luua mobiilsetel seadmetel kasutamiseks ettenähtud restoranide veebirakendus, mis lihtsustab teenindajate tööd ning hoiab tööle kuluvat aega kokku. Samuti pakub see restoranikülastajatele kiiremat ja mugavamat viisi teenuse kasutamiseks, mis oma korda suurendab rakenduse kasutajate arvu. Rakendusega on kasutajal võimalik broneerida laud, sirvida menüüd, lisada valitud tooted ostukorvi, tellimuse eest maksta ja soovi korral jätta jootraha.

Eesmärkide täitmiseks uuriti juba olemasolevat mobiilirakendust ja selle ülesehitust. Enne rakenduse arenduse algust koostati rakenduse vooskeem ja olemi-suhte diagramm ning valiti välja kasutatavad töövahendid.

Tagarakendus realiseeriti .NET 5.0 platvormil kasutades C# programmeerimiskeelt ja eesrakendus Reacti teegil. Kasutajaliidese kujundamiseks ja realiseerimiseks kasutati Material-UI komponenditeeki, päringute ja rakenduse oleku salvestamiseks React Query teeki. Rakenduse arendus toimus kahenädalaste sprintide kaupa ning rakenduse jupp-juppiti valmimist esitleti pidevalt kliendile, kes andis tagasisidet ning pakkus välja endapoolseid ettepanekuid muudatusteks ja funktsionaalsuse lisamiseks.

Pärast rakenduse valmimist anti kliendile võimalus seda testida, et olla kindel rakenduse töökindluses, kasutajakogemuses ja vigade puudumises. Vastava tagasiside pealt sai teha viimased muudatused, enne kui rakendus kõigile kasutatavaks tehti. Viimase hetkeseisuga pole kliendi sõnul ühtegi probleemi tekkinud ning restoranid on valminud rakendusega rahul. Sellega võib öelda, et töö alguses püstitatud eesmärgid said täidetud.

Kasutatud kirjandus

[1] R. Barger, „Is React a Library or a Framework? Here's Why it Matters“ 2021. [Võrgumaterjal]. Saadaval: https://www.freecodecamp.org/news/is-react-a-library-or-a-framework/. [Kasutatud 13.11.2021].

[2] React, „Getting Started“ [Võrgumaterjal]. Saadaval: https://reactjs.org/docs/getting-started.html. [Kasutatud 14.11.2021].

[3] M. Manjunath, „AngularJS and Angular 2+: a Detailed Comparison“ 2018. [Võrgumaterjal]. Saadaval: https://www.sitepoint.com/angularjs-vs-angular/. [Kasutatud 14.11.2021].

[4] Vue, „Introduction“ [Võrgumaterjal]. Saadaval: https://vuejs.org/v2/guide/. [Kasutatud 14.11.2021].

[5] V. Cromwell, „Evan You“ 2016. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/. [Kasutatud 14.11.2021].

[6] R. Barger, „Is React a Library or a Framework? Here's Why it Matters“ 2021. [Võrgumaterjal]. Saadaval: https://www.freecodecamp.org/news/is-react-a-library-or-a-framework/. [Kasutatud 14.11.2021].

[7] Vue, „The Progressive JavaScript Framework“. [Võrgumaterjal]. Saadaval: https://vuejs.org/. [Kasutatud 14.11.2021].

[8] I. Codesido, „What is front-end development?“ 2009. [Võrgumaterjal]. Saadaval: https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost. [Kasutatud 14.11.2021].

[9] PostgreSQL, „About“. [Võrgumaterjal]. Saadaval: https://www.postgresql.org/about/. [Kasutatud 14.11.2021].

[10] M. Asay, „There's one big reason that Postgres can't kill Oracle, and it's not the technology“ 2017. [Võrgumaterjal]. Saadaval: https://www.techrepublic.com/article/theres-one-big-reason-that-postgres-cant-kill-oracle-and-its-not-the-technology/. [Kasutatud 14.11.2021].

[11] Spring, „Spring Boot“. [Võrgumaterjal]. Saadaval: https://spring.io/projects/spring-boot. [Kasutatud 14.11.2021].

[12] Oracle, „What is a Relational Database (RDBMS)?“. [Võrgumaterjal]. Saadaval: https://www.oracle.com/database/what-is-a-relational-database/. [Kasutatud 14.11.2021].

[13] Amazon, „What is a Relational Database?“. [Võrgumaterjal]. Saadaval: https://aws.amazon.com/relational-database/. [Kasutatud 14.11.2021].

[14] Computer Weekly, „Write once, run anywhere?“ 2002. [Võrgumaterjal]. Saadaval: https://www.computerweekly.com/feature/Write-once-run-anywhere. [Kasutatud 14.11.2021].

[15] Microsoft [Võrgumaterjal]. Saadaval: https://dotnet.microsoft.com/platform/why-choose-dotnet. [Kasutatud 15.11.2021]

[16] Microsoft, 2021 [Võrgumaterjal]. Saadaval: https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends. [Kasutatud 28.11.2021]

[17] R. Abdul Rasheed, „Why “Backend For Frontend” Application Architecture?“ [Võrgumaterjal]. Saadaval: https://www.mobilelive.ca/blog/why-backend-for-frontend-application-architecture/. [Kasutatud 28.11.2021]

[18] React, „File Structure“ [Võrgumaterjal]. Saadaval: https://reactjs.org/docs/faq-structure.html. [Kasutatud 28.11.2021]

[19] Vipps, „About Us“ [Võrgumaterjal]. Saadaval: https://vipps.no/om-oss/. [Kasutatud 28.11.2021]

[20] R. de Best, „Share of users of mobile payment apps in Norway 2019“, 2021 [Võrgumaterjal]. Saadaval: https://www.statista.com/statistics/1098533/share-of-users-of-mobile-payment-apps-in-norway/. [Kasutatud 28.11.2021]

[21] 9000store, „ISO 9001 Flowchart Basics“ [Võrgumaterjal]. Saadaval: https://the9000store.com/articles/iso-9000-tips-iso-9001-flowchart-basics/. [Kasutatud 28.11.2021]

[22] NUnit, „What Is NUnit?“ [Võrgumaterjal]. Saadaval: https://nunit.org/. [Kasutatud 03.12.2021]

[23] Microsoft, „.NET Programming Languages“ [Võrgumaterjal]. Saadaval: https://dotnet.microsoft.com/languages. [Kasutatud 03.12.2021]

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

Mina, Indrek Kann

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Mobiilsetel seadmetel kasutamiseks ettenähtud restoranide veebirakenduse arendamine“, 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.

04.01.2022

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.