TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Viktoria Mihhailova 206758IADB

Toitlustusasutuste klienditeeninduse automatiseerimise prototüüp

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

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: Viktoria Mihhailova

13.05.2023

Annotatsioon

Käesoleva diplomitöö eesmärgiks oli luua veebirakenduse prototüüp, mis on mõeldud ennekõike toitlustusasutuste automatiseerimiseks ning nende tööd effektiivsemaks muutmiseks.

Loodi veebirakenduse prototüüp, mis võimaldab asutuse kliendil tellida toitu, broneerida lauda ja maksta rakenduse kaudu ning saada soodustusi. Asutuse omanikule rakendus võimaldab menüü muutmist ning ürituste korraldamist. Peale ürituste saab omanik ka korraldada igapäevaseid külastusi toitlustusasutusse.

Arendus koosnes kolmest etappist, taga- ja eesrakenduse arendamine ja testimine. Antud töö analüütiline osa annab ülevaadet olemasolevatest tehnoloogiatest ja lahendustest ning seletab arenduses kasutatava arhitektuurilist mustrit.

Arenduse tulemuseks on töötav veebirakenduse prototüüp. Töös on väljatoodud ka edaspidine plaan arendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 55 leheküljel, 6 peatükki, 26 joonist, 2 tabelit.

Abstract

Customer Service Automation Prototype for Establishments Providing Food

The aim of this thesis is to create a prototype of a web application, which is primarily intended for the automation of establishments providing food and making their work more effective.

A created prototype of the application, allows the establishment's customer to pre-order food, book a table, pay through the application and receive discounts. The application gives the owner of the establishment the possibility of adding a menu and organizing events. In addition to events, the owner can also arrange daily admission to the establishment through the app. The development consists of three stages, testing and development of the back-end and front-end applications. The analytical part of this work provides an overview of existing technologies and solutions and explains the architectural pattern used in the development.

The result of the development is a working web application prototype. The work also outlines the future plan for the development of the prototype.

The thesis is in Estonian language and contains 55 pages of text, 6 chapters, 26 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* ehk rakenduse programmeerimise liides
COVID-19	*Coronavirus Disease* 2019
CRUD	Lühend *create*, *read*, *update*, *delete* ehk loomise, lugemise, värskendamise ja kustutamise funktsioonite kirjeldamiseks
CSS	*Cascading Style Sheets* ehk veebilehtede standardne stiilikeel
Dependency Injection	Sõltuvuste süstimine
DOM	*Document Object Model* ehk dokumendi objektimudel
Domain Layer	Domeenikiht
ERD	*Entity Relationship Diagram* ehk olemi-suhte diagramm
Footer	Jalus
Header	Päis
HTML	*HyperText Markup Language* ehk veebilehtede standartne märgistuskeel
HTTP	*Hypertext Transfer Protocol* ehk protokoll andmetevahetuseks võrgus
IKT	Info-ja kommunikatsioonitehnoloogia
JSON	*JavaScript Object Notation*, andmetevahetuse formaat
Mapper	Teisaldusklass
Onion Architecture	Sibularhitektuur on tarkvaraarhitektuur, mida kasutatakse veebirakenduste loomiseks
Presentation Layer	Esitluskiht
QR-kood	*Quick Response Code* ehk ruutkood
Repository Layer	Hoidla kiht
REST API	*Representational State Transfer Application Programming Interface* ehk rakenduse programmeerimise liidese arhitektuuriline stiil
Service Layer	Teenusekiht
SQL	*Structured Query Language* ehk struktureeritud päringukeel
Unit	Ühik

User Happy Path	Optimaalne kasutajakogemuse tee

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva kiires ja tehnoloogiapõhises maailmas on automatiseerimine muutunud paljude tööstusharude oluliseks aspektiks, sealhulgas toitlustussektoris. Viimastel aastatel on toitupakkuvate asutuste omanikud teinud olulisi muudatusi, et kohaneda tarbijate muutuvate eelistustega tehnoloogias. Tehnoloogiate kasutamine on muutnud inimeste toidu tellimise ja selle eest maksmise viise, kuna paljud asutused pakuvad mobiilset tellimist, viipemakseid ja muud. Siiski on endiselt palju toitlustusasutusi, mis ei kasuta rakendusi ega tarkvara oma töö erinevate aspektide automatiseerimiseks. See võib põhjustada mitmesuguseid probleeme, nagu ebatõhusus, täpsuse vähenemine, seega vigade suurenemine ja tootlikkuse vähenemine. Näiteks käsitsi tehtavad protsessid, nagu tellimuste vastuvõtmine, laoseisu jälgimine ja töötajate ajakava koostamine, võivad olla aeganõudvad ja altid vigadele, mille tulemusena väheneb tõhusus ja suurenevad kulud.

Lisaks võib automatiseerimise puudumine kliendikogemust negatiivselt mõjutada, kuna käsitsi tehtavad protsessid võivad pikendada ooteaega ja suurendada klientide pettumust. Tänapäeva konkurentsitihedal turul on klientide meelitamiseks ja hoidmiseks ülioluline sujuva ja tõhusa kliendikogemuse pakkumine.

Paljud toitlustusasutused lisaks toitlustusteenuste osutamisele korraldavad ja viivad läbi üritusi näiteks lauamängude või elava muusika õhtud. Kasutades ürituste korraldamiseks rakendusi ja tarkvara saab erinevaid ülesandeid automatiseerida ja sujuvamaks muuta võimaldades ürituste korraldajatel keskenduda ürituse muudele olulistele aspektidele. Näiteks võib ajaplaneerimstarkvara automatiseerida broneerimise ja ajastamise protsessi, vähendades topeltbroneeringu või ajakavaga seotud konfliktide ohtu. Samamoodi saab eelarvestamise tarkvara automatiseerida kulude jälgimise ja aruannete koostamise ning statistika kogumise protsessi, muutes eelarve haldamise lihtsamaks ja tagades, et sündmus jääb eelarve piiridesse. Lisaks võib automatiseerimine parandada ka osalejate üldist kogemust, kuna see võib sujuvamaks muuta eelregistreerimise, muutes osalejate jaoks üritusel osalemise lihtsamaks ja meeldejäädavamaks.

2 Probleemi ülevaade

Järgnevas peatükkis tuuakse välja püstitatud probleemi lahendamiseks valitud metoodikat, vaadeldakse probleemidele erinevaid lahendusi. Võrreldakse olemasolevaid IKT ehk info-ja kommunikatsioonitehnoloogia süsteeme ning analüüsitakse nende puudujääke. Pakutakse välja uus lahendus ja selgitatakse selle eeliseid teiste rakenduste seast. Tuuakse välja käsitletava töö skoop.

2.1 Metoodika

Käsitletavas lõputöös analüüsitakse püstitatud probleemi Eesti kontekstis. Võrreldakse olemasolevaid analüüsitava probleemi lahendusi, mis tänapäeval Eestis kasutusel. Tuuakse välja nende tugevad ja nõrgad küljed ning pakutakse välja uus lahendus, mis lahendab probleemi.

Töö analüütilises osas probleemi seletatakse lahti funktsionaalseteks nõueteks, mille alusel tuuakse välja kasutajalood. Vaadeldakse erinevaid tehnoloogiaid, mis sobivad käsitletava lahenduse loomiseks ning analüüsi käigus valitakse sobivaimad.

Rakenduse projekteerimisel luuakse olemi-suhte diagramm andmebaasi disaini kirjeldamiseks ning kasutajaliidese disaini prototüüp. Töö käigus valmib veebirakenduse prototüüp, mis koosneb taga- ja eesrakendusest. Viiakse läbi ka valminud veebirakenduse testimist, mis koosneb tarkvara ja kasutajakogemuse testimisest.

2.2 Eksisteerivad lahendused

Eestis on kasutusel mitut rakendust, mis aitavad toitlustusasutustel enda tööd lihtsustada ja kiirendada. Eesti elanikud kasutavad pakutavaid tehnoloogilisi lahendusi aktiivselt. Uuringud näitavad, et pea 70% Tallinna ja Harjumaa elanikest on oma elus kasutanud e-poe teenust [1].

Olemasolevad rakendused, mingil määral lahendavad käsitletavas töös püstitatud probleeme, kuid ei lahenda neid täielikult. Vaadeldakse kolme tüüpi lahendusi, millel on erinevad eesmärgid, mis tegelevad toiduasutustega või ürituste korraldamisega.

2.2.1 Toidu kohaletoimetamine

Toidu kohaletoimetamine on muutunud üha populaarsemaks. See annab inimestele võimaluse nautida restoranikvaliteediga toite oma kodust väljaminemata. Umbes 42 protsendi Eesti elanikest tellivad vähemalt kord kuus toitu koju [2]. Eriti *COVID-19* pandeemiaga muutus antud teenus veel kasutatavamaks. Toidu kohaletoimetamise teenuste kasutamine aitab kliendile planeerida aega ning ajaliselt jälgida protsessi. Samas asutusele, mis on liitunud antud teenusega, on lihtsam koguda statistikat ning vähendada kulusid, mis on seotud klienditeenindusega, kuna kliendi teenindamine toimub mitte füüsiliselt vaid rakenduse kaudu.

Eestis tuntumad ning enamkasutatud toidu kohaletoimetamise teenust pakkuvad lahendused on *Bolt Food*, *Wolt* ja *Fudy*. Edasi analüüsitakse ühe neist, kuna analüüsi käigus selgus, et rakenduste funktsionaalsus on sarnane ning erineb vaid hindade ning kasutajaliidese poolest. Seega otsustati vaadelda *Bolt Food* kuna see on enim kasutatud lahendus Eestis.

Bolt Food on kohaletoimetamisplatvorm, mis pakub kiireid ja kvaliteetseid toidu kohaletoimetamise teenuseid taskukohaste hindadega [3]. *Bolt Food* on lisateenus, mis kuulub Eesti firmale *Bolt*. Eestis avati Bolt Food toidutellimise võimalus 2019 aasta suvel [4]. Toitu pakkuva asutuse juht võib liita enda restorani *Bolt Food* rakendusega täides ankeedi veebilehel [5]. Liitunud asutus on rakenduse esilehel kõikide restoranide nimekirjas. Toidu tellimiseks rakenduse kaudu on vaja luua konto, lisada maksmiseviis (pangakaart või sularaha) ning määrata enda asukoha kuhu toitu soovitakse tellida. Vajutades ühele restoranile nimekirjast, avaneb valitud restorani menüü. Vastavalt vajadustele kasutaja saab lisada menüüs olevaid tooteid enda ostukorvi ning vormistada tellimus. Pärast makse õnnestumist saab klient jälgida enda tellimuse valmistamise etappe ning esialgset plaanitava toimetamisaega.

Bolt Food ning sarnased rakendused annavad liitunud asutustele palju eeliseid. Liitumine aitab jõuda uute kasutajateni, suurendada enda läbimüüki ning keskenduda rohkem enda ärile, kuna platvorm võttab enda peale toimetamise ja logistikaga seotud probleemid [5]. Ühelt poolt rakendus aitab ka koguda statistikat ning vähendada seotud tööjõuga kulusid, kuid mitte täielikult. Kuna *Bolt Food*, *Wolt* või *Fudy* rakenduse kaudu tellib toidu vaid osa klientidest siis endiselt suur osa ei kasuta rakendust. Kliendil on võimalus tellida toidu kaasa ja tulla ise selle järgi, kuid ei ole võimalust tellida ja süüa kohapeal restoranis.

2.2.2 Ürituste korraldamine

Ürituse korraldamine võtab juhil palju aega, kuna on vajalik läbimõelda paljud aspektid: piletite müük, sissepääsu kontroll ja ülebroneerimise vältimine. Turul on ettevõteid, mis pakuvad ürituste korraldamise teenuseid, kuid antud töö raames on oluline leida neid, kes pakuvad tarkvara, mida saab kasutada ürituste korraldamiseks, näiteks *Fienta* ja *Mitto Events*.

Fienta on ettevõte, mis pakub ürituse korraldajale tarkvara, mis automatiseerib ning seetõttu aitab lihtsustada piletimüügi, osalejate registreerimist, maksete vastuvõtmist ning automaatset arveldust [6]. Platvormiga liitumiseks peab ettevõte juht registreerima ennast ürituse korraldajana veebilehel ning saatma taotluse ürituse korraldamiseks. Taotluse kinnitamisel luuakse tellitud üritusele eraldi veebileht, mida korraldaja saab muuta, lisada pilte ning valida lehe kujunduses kasutatavaid värve. Ürituse lehelt saavad huvilised osta sissepääsu pileti, mida hiljem valideeritakse kas *QR-koodina* (*Quick Response Code*) ehk ruutkoodina või dokumendifailina [7].

Finta on mugav lahendus ürituste korraldamiseks, mis aitab korraldajal aega säästa. Eriti sobilik antud ettevõte teenus on suuremate ürituste korral, kuna erinevate protsesside jälgimine võib muutuda võimatuks ilma ühtse süsteemita, mis aitab vältida vigu. Käsitletavas töös vaadeldakse toitu pakkuvaid asutusi, mis võiksid kasutada ülalnimetatud ettevõte teenusi, kuid tihtipeale sellistes asutustes toimuvad üritused on väiksed ning ei nõua suure osalejate arvu arvestamist. Näiteks baarid, kus erinevad üritused võivad toimuda kaks ja enam korda nädalas. Iga sellise ürituse korraldamisel, teise ettevõte teenuse ostmine on kulukas töö.

2.2.3 *Catering*

Catering ehk toitlustusteenus on toidu ja joogide pakkumine suurele hulgale inimestele teatud ürituse raames, näiteks pulmadel, peodel ja teistel üritustel [8]. Antud uurimustöö raames catering on vaadeldamiseks oluline, kuna see pakkub tellijale ürituse ajaks toidu tellimise võimaluse. Seda teenust saab mõnel määral võrrelda pakutava lahendusega, kuna toidu kohaletoimetamise aeg on ettemääratud ning ettearvatud. Eestis on mitut ettevõtet, mis pakkuvad toitlusteenuseid. Järgnevalt vaadeldakse ühe neist, kuna suuremas osas ettevõted pakkuvad sarnast teenust.

AB Catering on 2012. aastal loodud ettevõte, mis pakkub catering teenuseid Eestis [9]. Teenuste tellimiseks on vaja täita veebilehel vorm enda kontaktandmetega ning *AB Catering* töötajad ise võttavad tellijaga ühendust. Ettevõte aitab tellijal arvestada toidu kogust inimeste arvuga ning toidu valikuga. Antud teenus on mõeldud suurematele üritustele ning selle töö ei ole korraldatud tarkvara abiga. Seega olukorras kui toitu on arvestatud teatud inimeste arvuga ning mingi osa ei tule üritusele kohale tekkivad ülejäägid, mis põhjustavad üleliigseid kulutusi.

2.2.4 Olemasolevate lahenduste võrdlus

Järgnevas tabelis tuuakse välja eelnevalt analüüsitud kolm kategooriat ning võrreldakse igasse kategooriasse kuuluvaid lahendusi käsitsetavas töös pakkutava lahendusega. Võrdluskriteeriumid, tabelis veerg „Süsteemi funktsionaalsus“ on valitud võetes neid kokku, nii et see funktsionaalsus lahendaks töös püstitatud probleemi. Probleemi lahendamiseks peaks rakendus koguma statistikat, andma kliendile toidu broneerimise võimaluse, sobima ürituste korraldamiseks ja sellele piletite müügiks.

Tabel 1. Olemasolevate lahenduste võrdlus.

Süsteemi funktsionaalsus	Toidu kohaletoimetamine	Ürituste korraldamine	*Catering*	Uus lahendus
Statistika kogumine	Jah	Jah	Ei	Jah
Toidu broneerimise võimalus	Ei	Ei	Jah	Jah
Sobilik üritustele	Ei	Jah	Jah	Jah
Üritusele piletite müügi võimalus	Ei	Jah	Jah	Jah

Kõik kolm kategooriat on omavehel väga erinevad ning võrrelda neid on raske, kuid tabelist 1 saab näha, et analüüsitavad rakendused on suunatud ühele konkreetsele eesmärgile. Eesmärk on kas toidu kohaletoimetamine või teatud ürituse korraldamine. Seega uus lahendus ühendaks kaks aspekti endas kokku.

2.3 Lahenduse skoop

Antud töö raames luuakse uue lahenduse prototüüp ning kirjeldatakse erinevaid lähenemisviise ja tehnoloogiaid, mis analüüsi käigus selgusid parimaks antud töö või olukorra jaoks. Luuakse veebirakenduse prototüüp põhilise kirjeldatud funktsionaalsusega, mis lahendab töös käsitletavaid probleeme. Tuuakse välja ka edaspidised plaanid rakenduse arendamiseks funktsionaalsuse poolelt.

Prototüübi valmimisel viib töö autor läbi sihtgruppi abil testimise, mis aitab saada tagasisedet rakenduse prototüübi kasutamise kohta.

2.3.1 Pakutav lahendus

Töö käigus valmiv veebirakendus on eelkõige mõeldud kui konkreetsele asutusele pakutav infosüsteem. Asutustel, mis liituvad pakutava lahendusega, on enda oma anbmebaas, kus hoitakse kõik sellega seotud andmed. Seega rakendus on iga konkreetse kliendi jaoks muudetav, sealhulgas ka süsteemi funktsionaalne osa ja kasutajaliidese disain.

Loodava lahenduse eesmärk on eelkõige pakkuda võimalust asutuse omanikul vähendada kulusid töö korraldamise osas. Seega uue lahenduse põhiline funktsionaalsus on veebirakenduse kaudu laua broneerimine ning toidu teatud ajaks tellimine. Tellimuse eest maksmine toimub rakenduse kaudu, mis annab võimaluse ooteaega vähendada või isegi vältida. Veebirakendus toetab ka soodustuste süsteemi. Erinevatel omaniku poolt määratud tingimustel, rakenduse kasutaja saab endale soodustusega kuponge ning neid võib klient aktiveerida järgmistel ostudel.

Pakutav lahendus annab võimaluse asutusele korraldada üritusi, mille abil saab klient valida endale sobiva üritusele pileti, lisada selle endale ostukorvi ning pärast tellimuse vormistamist rakendus genereerib *QR-koodina* ehk ruutkoodina pileti.

3 Pakutava lahenduse analüüs

Antud peatükkis vaadeldakse olemasolevaid tehnoloogiaid, mis sobivad antud töös püstitatud probleemi lahenduseks ning valitakse veebirakenduse arendamiseks sobivaimad.

Selleks, et selgitada kuidas pakutav lahendus töötab, kes võiks seda kasutada ja mis eesmärgil tuuakse välja veebirakendusele nõuded ning kirjutatakse need välja kasutajaloodena. Eraldi tuuakse välja nii käsitletava töö skoopi kuuluvaid nõuded kui ka neid mida täidetakse tulevikus. Nõudeid rakendusele vaadeldakse kahelt poolelt, kliendi ja asutuse omaniku pooltelt eraldi.

Väljatoodud nõuete abil tehakse valmis rakenduse visuaalne prototüüp, et anda ülevaadet loodavast rakendusest ning selle loodavast funktsionaalsust. Selle põhjal luuakse andmebaasi kohta olemi-suhte diagramm.

3.1 Nõuded ja kasutajalood

All on toodud välja nimekiri kasutajaloodest, mille põhjal luuakse nõuded uue rakenduse prototüübi jaoks. Nende koostamisel võeti aluseks keskmise restorani või kohviku tööpõhimõte - see tähendab, et lisaks peale uute funktsionaalsuste lisamise peab uus rakendus toetama põhilisi toitlustusasutuse tegevusi: laua broneerimine, toidu tellimine, maksmine. Töö autor toob välja nõudeid toetudes nii enda kogemusele, kui ka olemasolevate lahenduste võimalustele.

Kliendi nõuded rakenduse prototüübi funktsionaalsele:

- Kliendina soovin registreerida ennast kasutajaks.
- Kliendina soovin osta toidu läbi rakenduse.
- Kliendina soovin broneerida lauda.
- Kliendina soovin tellida toitu broneeritud lauale.
- Kliendina soovin tellida toitu broneeritud lauale ettemääratud ajaks.
- Kliendina soovin lisada enda pangakaardi rakendusele makseviisiks.

- Kliendina soovin näha millised lisatud pangakaardid on kehtivad.
- Kliendina soovin saada soodustusi ja kuuponge.
- Kliendina soovin osta üritustele pileteid.
- Kliendina soovin valideerida ostetud pileti läbi rakenduse.
- Kliendina soovin vaadata enda varem tehtud tellimusi.
- Kliendina soovin näha enda kasutajakonto taset.

Asutuse omaniku nõuded rakenduse prototüübi funktsionaalsele:

- Asutuse omanikuna soovin enda restoranile veebilehte.
- Asutuse omanikuna soovin lisada toidu kategooriad.
- Asutuse omanikuna soovin lisada uut toitu teatud kategooriasse.
- Asutuse omanikuna soovin lisada korraldatava ürituse piletid.
- Asutuse omanikuna soovin müüa korraldatava ürituse piletid.
- Asutuse omanikuna soovin lisada soodustuste ja kupongide saamise tingimusi.
- Asutuse omanikuna soovin näha tehtud tellimuste arvu.

3.2 Tulevikus loodava funktsionaalsuse nõuded ja kasutajalood

Rakenduse skoop on piiratud seega mõned nõuded täidetakse väljaspool käsitletava töö skoobi. All on välja toodud nimekiri kasutajaloodest, mida täidetakse tulevikus.

Kliendi nõuded rakenduse funktsionaalsele:

- Kliendina soovin jälgida tellimuse valmistamise protsessi.
- Kliendina soovin maksta läbi rakenduse.

Asutuse omaniku nõuded rakenduse funktsionaalsusele:

- Asutuse omanikuna soovin näha teatud toodete ostmise statistikat.

- Asutuse omanikuna soovin näha asutuse külastuste statistikat.
- Asutuse omanikuna soovin vältida laudade ülebroneerimist.

3.3 Tehnoloogia valik

Valmistatava veebirakenduse arendamiseks on oluline valida sobivat andmebaasimootori. Rakendus koosneb taga-ja eesrakendusest seega on tähtis ka analüüsida olemasolevaid raamistike ja tehnoloogiaid, mis sobivad arenduseks ning hiljem ka rakenduse haldamiseks.

Rääkides tehnoloogiate valikust on olulisem vaadelda mitte programmeerimiskeelt vaid raamistiku, mida saab arenduses kasutada. Programmeerimiskeel on käskude kogum, mis juhendab tarkvara või arvuti funktsionaalsust. Raamistik aga loob aluse keele käskude rakendamiseks [10]. Valides sobiva programmeerimiskeele arenduseks ei pruugi antud keelele üles ehitatud raamistik olla sobilik.

3.3.1 Andmebaasimootor

Andmebaas on andmete järjestatud kogum, mis tavaliselt salvestatakse arvutisüsteemi mällu. Kaasaegsete levinumate andmebaasidesse andmed salvestatakse tavaliselt ridade ja veergude kujul, mis moodustavad tabeli. Neid andmeid saab hallata, muuta, ajakohastada, jälgida ja korrastada. Enamik andmebaase kasutab andmete kirjutamiseks ja päringute tegemiseks struktureeritud päringukeelt *SQL* (*Structured Query Language*) [11].

Maailmas on erinevaid tüüpe andmebaase: relatsioonilised, mitterelatsioonilised, objekt-orienteeritud, puu-kujulised, hajutatud, vektor. Igal andmebaasi tüübil on omad eelised ja nõrkused. Kuna töö käigus arendatakse just veebirakendust, tehakse valik relatsiooniliste andmebaaside kasuks kuna töö autoril on relatsiooniste anbmebaasidega rohkem kogemusi.

Relatsiooniandmebaasi võib kasutada mis tahes teabevajaduse jaoks, milles andmepunktid on üksteisega seotud ning mida tuleb hallata turvalisel, reeglipõhisel ja järjepideval viisil. Relatsioonimudeli eelised teevad sellest jätkuvalt andmebaaside

enimkasutatud mudeli [12]. Alljärgnevalt on välja toodud tänapäeva tuntumad relatsioonilised andmebaasid.

PostgreSQL on vabavaraline võimas andmebaas, millel on suur kasutajate baas, seega kiiresti saab leida lahendusi arenduse käigus tekkinud probleemidele. *PostgreSQL* toetab ka piltide, häälte ja videote salvestamist [13] [14].

MySQL on vabavaraline andmebaas. *MySQL* tagab turvalisust, usaldusväärsust, kuid litsents ei ole tasuta, kui andmebaasi kasutatakse ärilistel eesmärkidel. Samas tasuta versiooni funktsionaalsus on piiratud [15].

MariaDB on vabavaraline andmebaas, mis on arendatud *MySQL* andmebaasi põhjal. MySQL andmebaasist erineb *MariaDB* sellega, et funktsionaalsus ei ole piiratud ega jagatud erinevate litsentside vahel. Samas *MariaDB* on kiirem [16].

SQLite on vabavaraline ning tasuta. Andmebaas pakub suurt töökindlustust ning täsfunktsionaalsust. *SQLite* on ka maailmas enimkasutatud andmebaasimootor, kuid selle kasutatava mälu suurus ning funktsionaalsus on limiteeritud [17].

Eelnevalt välja toodud erinevate andmebaaside omaduste põhjal antud töö veebirakenduse prototüübi arendamiseks valitakse *PostgreSQL* andmebaasimootori, kuna see on tasuta, pakub suurt võimsust, kiireid päringuid ning sobib veebirakenduse arendamiseks. Lisaks on töö autoril *PostgreSQL* andmebaasiga rohkem kogemusi.

3.3.2 Tagarakenduse programmeerimiskeel ja raamistik

Tagarakendus vastutab serveripoolse rakenduse eest. See tähendab, et tagarakendus suhtleb andmebaasiga ning töötleb saadetud andmeid läbi rakenduse programmeerimise liidese ehk *API* (*Application Programming Interface*) [18]. Loodav veebirakendus peaks toetama kasutajate autentimist, autoriseerimist.

Järgnevalt on välja toodud enimtuntumad programmeerimise raamistikud:

Spring on *Java* programmeerimiskeelel põhinev raamistik, mis on tasuta ning ei nõua litsentsi. *Spring* toetab veebirakenduse arendamist ning pakub sellele tugi suure

dokumentatsiooni abil. Lisaks *Spring* toetab sõltuvuste süstimist (*dependency injection*), rakenduse testimise võimalusi ning mooduleid, lihtsustavaid liideseid andmebaasiga suhtlemiseks [19]. Lihtsustamaks *Spring* raamistiku kasutamist, loodi *Spring Boot*, mis on omakorda raamistik. See pakub kiiremat projekti konfigureerimist, kuid laeb projekti suure arvu lisamooduleid, mis ei pruugi olla kasutatud, seega projekti suurus kasvab [20].

.NET on *Microsoft´*i poolt loodud *C#* programmeerimiskeelel põhinev vabavaraline raamistik rikka dokumentatsiooniga. Raamistik pakub palju võimsaid arendustööriistu, mis muudavad protsessid kiiremaks. Kirjutatud koodi on lihtne hooldada ja siluda. *.NET Core* on uusim *.NET* raamistiku versioon, mis on arendatud mitte üle vana versiooni, vaid uuesti, selleks, et muuta seda kergemaks, kiiremaks ja platvormiüleseks raamistikuks. See toetab mikroteenuste arendamist, *REST API* (*Representational State Transfer Application Programming Interface*) ehk rakenduse programmeerimise liidese arhitektuuriline stiil, mis eeldab *HTTP* (*Hypertext Transfer Protocol*) protokolli päringute kasutamist ja palju muud [21] [22] [23].

Django on tasuta vabavaraline *Python* programmeerimiskeelel põhinev raamistik. Antud raamistiku on lihtne õppida, kuna see on kasutajasõbralik ning see aitab luua rakendusi kiiremini, kirjutades vähem koodi. *Django* pakub samuti laia dokumentatsiooni koos juhenditega [24].

Express.js on *JavaScript* programmeerimiskeelel põhinev raamistik, mis on arendatud *Node.js* peal selleks, et muuta veebirakenduste arendamist lihtsamaks ja kiiremaks, kuna ees- ja tagarakendust saab samas programmeerimiskeeles kirjutada. *Express.js* on hea lahendus väikeste ja keskmise suurusega rakenduste jaoks, kuna see on kerge ja seda on väga lihtne rakendada. Raamistik toetab *REST API* loomist, reaalaja teenusi, näiteks vestluste rakendusi loomist. *Express.js* saab kasutada ka staatiliste failide serveerimiseks, paljude samaaegsete ühenduste haldamiseks [25].

Järgnevas tabelis on toodud välja eelnevalt nimetatud raamistike võrdlus, mille põhjal tehakse uue rakenduse arendamiseks raamistiku valiku. Võrreldakse nelja kriteeriumi järgi: kogemus programmeerimiskeelega, kogemus raamistikuga, on raamistik vabavara või ei ole ning võimsus. Kriteeriume valiti nii, et loodava veebirakendust oleks kiire ja

lihtne arendada, kuid selle töötamise võimsus ei kahaneks. Ning oluline on ka see, et tehnoloogia kasutamine ei tooks lisakulusid ning oleks vabavara.

Tabel 2. Tagarakenduse raamistike võrdlus.

Raamistik	Kogemus programmeerimiskeelega	Kogemus raamistikuga	Vabavara	Võimsus
Spring (*Java*)	Hea	Keskmine	Jah	Väga hea
.NET (*C#*)	Hea	Väga hea	Jah	Väga hea
Django (*Python*)	Keskmine	Puudub	Jah	Hea
Express.js (*JavaScript*)	Keskmine	Puudub	Jah	Keskmine

Analüüsi käigus otsustati võtta kasutusele *.NET* raamistiku veebirakenduse tagarakenduse arendamiseks. Sellel on lai dokumentatsioon ning suur valik funktsionaalsuseid, mis toetavad ja lihtsustavad arendamist ning töö autoril on *.NET* raamistikuga rohkem kogemusi, võrreldes teiste raamistikega.

3.3.3 Tagarakenduse arhitektuur

ASP.NET Onion Achitecture ehk sibularhitektuur on tarkvaraarhitektuuri muster, mida kasutatakse veebirakenduste loomiseks. See põhineb ideel eraldada rakenduses probleemid, kihistades rakenduse komponendid.

Sibula arhitektuur koosneb neljast kihist:

Domain Layer ehk domeenikiht esindab rakenduse põhilist loogikat. See määratleb väärtusobjektid, mis juhivad rakenduse käitumist. See kiht on teistest kihtidest täiesti sõltumatu ja sisaldab rakenduse kõige olulisemat loogikat. Domeenikihti saab edasi jagada alamkihtideks, nagu olemid, väärtusobjektid ja domeeniteenused [26].

Repository Layer ehk hoidla kiht vastutab andmete püsivuse haldamise eest rakenduses. See annab andmetele juurdepääsukihi abstraktsiooni, võimaldades rakendusel lülituda

erinevate andmeallikate vahel, ilma et see mõjutaks ülejäänud rakendust. Hoidla kiht suhtleb domeenikihiga, et teha äriolemitel *CRUD* (*create*, *read*, *update*, *delete*) ehk loomise, lugemise, värskendamise ja kustutamise toiminguid [26].

Service Layer ehk teenusekiht esindab rakenduse põhilist äriloogikat ja sisaldab teenus klasse, väärtusobjekte ja ärireegleid [26].

Presentation Layer ehk esitluskiht vastutab kasutajaliidese ja kasutaja interaktsioonide haldamise eest. See sisaldab selliseid komponente nagu kontrollerid, vaated ja mudelid [26].

Sibulaarhitektuur on loodud nii, et kihtide vahel oleksid vead rangelt eraldatud. Iga kiht sõltub ainult vahetult selle all olevast kihist ja rakenduse tuumas olevad kihid on muust programmist täielikult eraldatud. See lähenemisviis võimaldab paindlikku, skaleeritavat ja hooldatavat rakenduse disaini. Samuti võimaldab see arendajatel hõlpsasti eri kihtide komponente välja lülitada, ilma et see mõjutaks rakenduse üldist struktuuri [26].

3.3.4 Eesrakenduse programmerimiskeel ja raamistik

Eesrakendus vastutab kasutaja ja tagarakenduse suhtlemise eest. Eesrakendus võtab vastu andmeid kasutajalt ning saadab neid tagarakendusele. Eesrakenduse osa on ka visuaalsed aspektid, sealhulgas värvid, paigutus ja fondid [18].

JavaScript on programmeerimiskeel, mis tänapäeval on enim populaarne eesrakenduste arendamisel ning antud lõputöö autoril on sellega kogemusi, seega antud peatükis analüüsitakse erinevaid *JavaScript´*i raamistike ning ei võta vaatluse alla teisi programmeerimiskeeli.

HTML (*HyperText Markup Language*) ja *CSS* (*Cascading Style Sheets*), mida kasutatakse veebilehe komponentide loomiseks ja disainimiseks ei ole programmeerimiskeeled. Need on vastavalt veebilehtede standardne märgistuskeel ehk komponentide kirjelduskeel ning stiilikeel. [27] Veebilehe disainimiseks otsustati võtta kasutusele *Bootstrap*. Bootstrap on maaimas enimkasutatud *CSS* klasside kogum, mis lihtsustab erinevate komponendide loomist ning disainimist [34].

*JavaScript´*i baasil on arendatud palju raamistike, mille seast valida. Alljärgnevalt on toodud välja mõned populaarsemad *JavaScript´*i teegid ja raamistikud:

React.js on *JavaScript´*i teek kasutajaliideste loomiseks. *React.js* on üks populaarsemaid *JavaScripti* teeke. Selle töötas välja ja haldab *Meta* ning üksikute arendajate ja ettevõtete kogukond, mille loodud lahendused on enamasti vabakasutatavad nii ärilisel eesmärgil kui ka mitte ärilisel [28]. *React.js* kasutab kasutajaliidese loomisel komponendipõhist lähenemist, mis muudab koodi taaskasutamise ja keerukate kasutajaliideste loomise kiiremaks ja lihtsamaks. *React.js* on tuntud ka oma suure jõudlusega tänu virtuaalse *DOM´*i (*Document Object Model*) ehk dokumendi objekimudeli kasutamisele, mis vähendab oleku muutumisel vaate uuesti renderdamiseks kuluvat aega ning lihtsustab oleku muutmist [29].

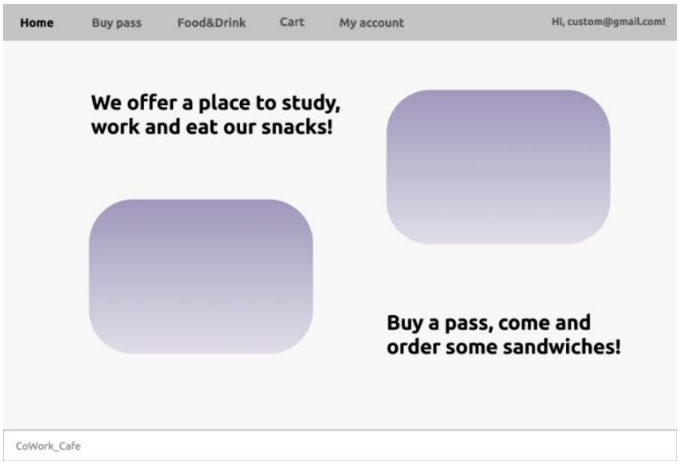
Vue.js on progressiivne *JavaScript´*i raamistik kasutajaliideste loomiseks, mille lõi ja haldab Evan You ja tema arendajate meeskond [28]. *Vue.js* sarnaneb *React.js´*iga selle poolest, et kasutab komponendipõhist lähenemist, kuid sellel on ka mõned olulised erinevused. Näiteks *Vue* malle on lihtsam lugeda ja kirjutada kui *React´*i mallikeelt. *Vue.js* raamistikul on lihtsam *API* kui *React.js´*il, mis muudab selle õppimise ja kasutamise algajatele lihtsamaks [30].

Angular.js on *JavaScript´*i raamistik veebirakenduste loomiseks. *Angular* kasutab komponendipõhist arhitektuuri, mis sarnaneb *React.js* ja *Vue.js´*iga, kuid sisaldab ka funktsionaalsust andmete sidumiseks, sõltuvuse sisestamiseks ja testimiseks. *Angular.js´*i on raskem õppida, kuid see pakub võimsaid tööriistu suuremahuliste rakenduste loomiseks. *Angular.js´*i kasutamiseks ei pea nii palju lisa pakette juurde laadima võrreldes *React´*i või *Vue´*ga, kuna enamust funktsionaalsust on juba raamistikus endas olemas [28].

Ülalnimetatud raamistikest valitakse *Vue.js*, kuna selle kasutamine arenduses on lihtsam nii andmete saamisel, kui ka nende töötlemisel ja suhtlemisel tagarakendusega, lisaks on autoril eelnevaid kogemusi eesrakenduste loomisel kasutades *Vue.js* raamistiku.

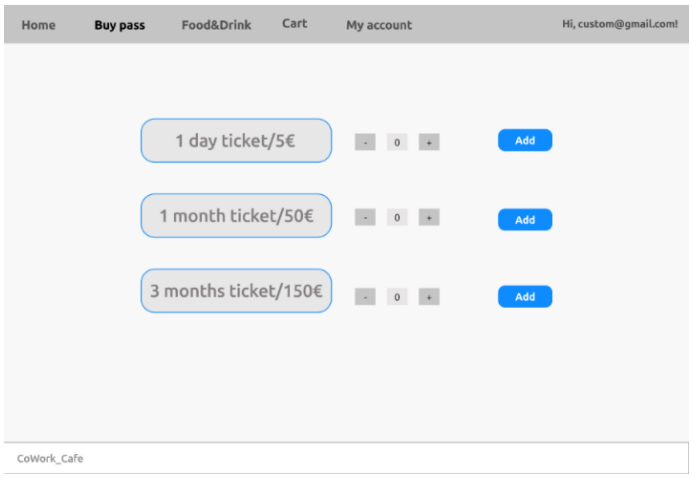
3.4 Lahenduse prototüüp

Enne rakenduse arendust, selleks et kujundada kuidas oleks paremini kasutajalugusid ning nõudeid täita, luuakse rakenduse kasutajaliidese visuaalne prototüüp. Antud lähenemisviis aitab hoida aega kokku, kuna kõik stsenaariumid mõeldakse läbi enne nende rakendamist ning arenduse käigus muudatused on minimaalsed. Käsitletava töö käigus loodava veebirakenduse visuaalse prototüübi loomiseks kasutatakse tuntud tarkvara *Figma*, mis pakub visuaalse prototüübi loomiseks kui ka testimiseks [31]. *Figma* tarkvara abil loodud visuaalne prototüüp annab vaid ettekujutuse tulevikust rakendusest. Loodav veebirakendus võib prototüübist erineda.



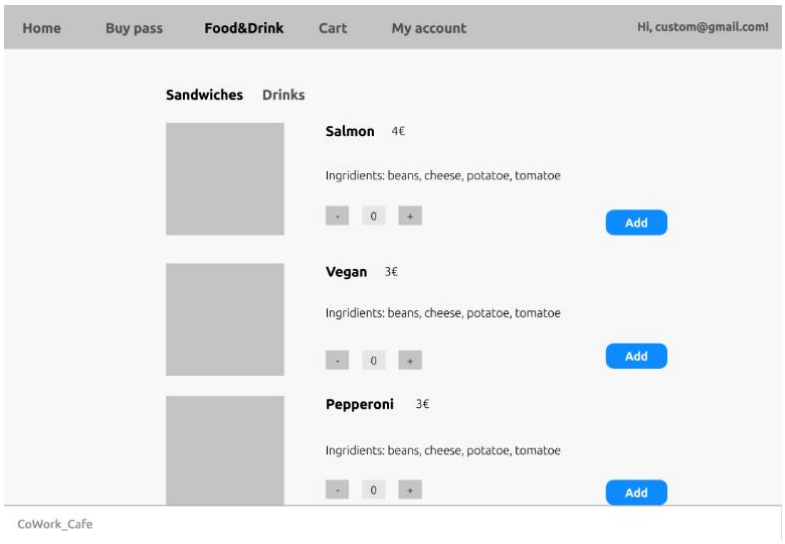
Joonis 1. Koduleht.

Joonisel 1 on näha uue veebirakenduse sisselogitud kasutaja esilehte. Lehel asutuse omanik võib paigaldada enda asutuse pilte ja reklaame näiteks korraldatavate ürituste infot.



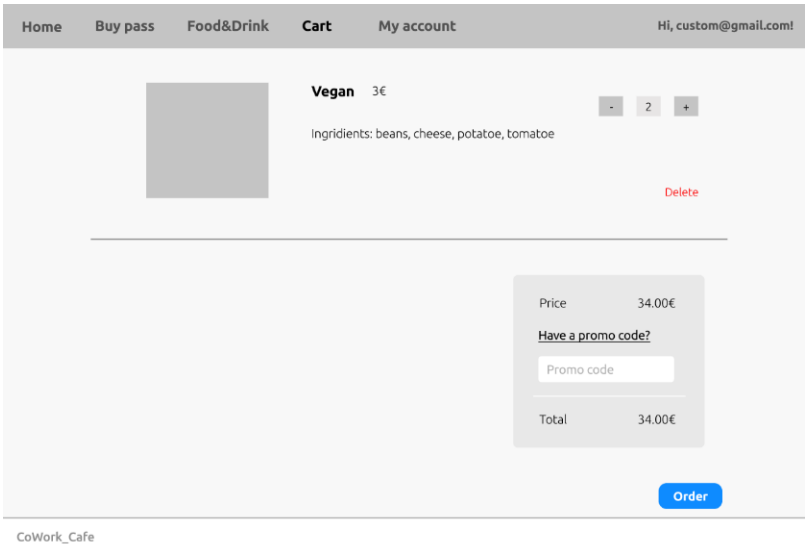
Joonis 2. Vaade piletite ostmiseks.

Joonisel 2 on kujutatud piletite ostmise vaade. Kasutaja saab lisada ostukorvi sobiva üritusele pileti vajutades „*Add*“ ehk „Lisa“ nuppule. Kasutaja võib valida ka piletite kogust vajutades „+“ või „-“ nuppudele.



Joonis 3. Vaade toidu kategooria menüüst.

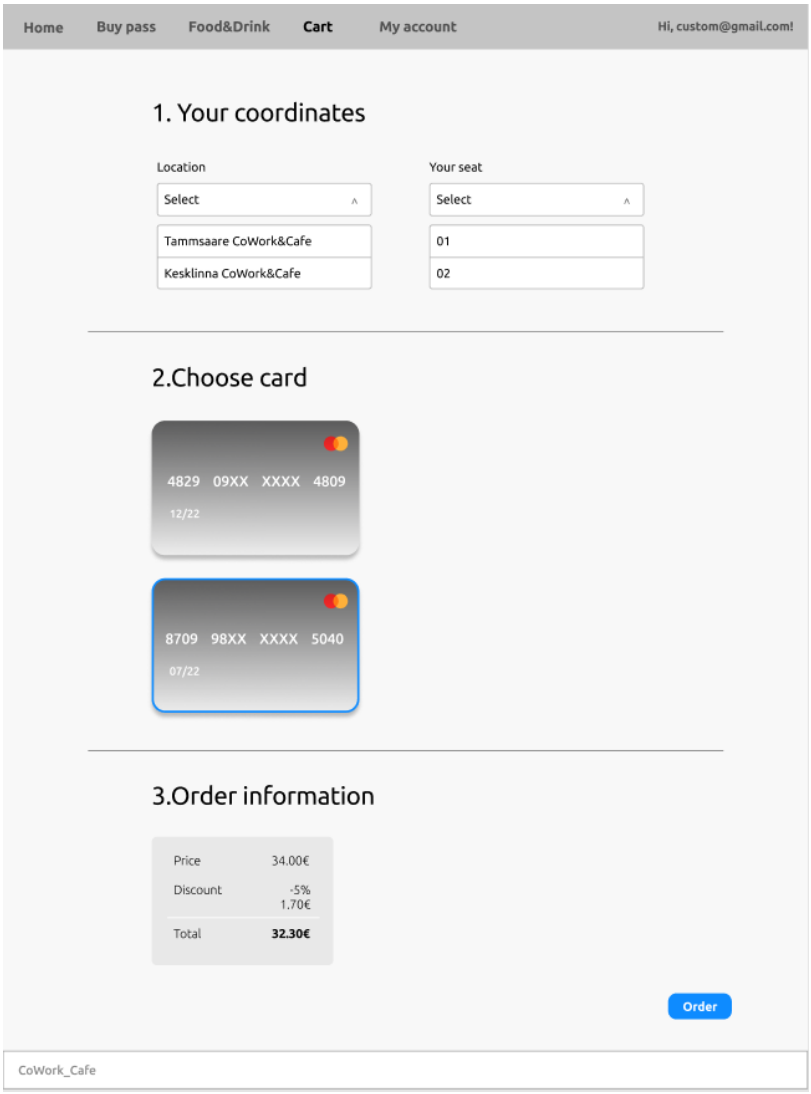
Joonisel 3 on kujutatud asutuse menüü kategooriate kaupa. Kasutaja saab valida sobiva toode ning selle koguse ja lisada enda ostukorvi.



Joonis 4. Ostukorvi vaade koos hinnaga.

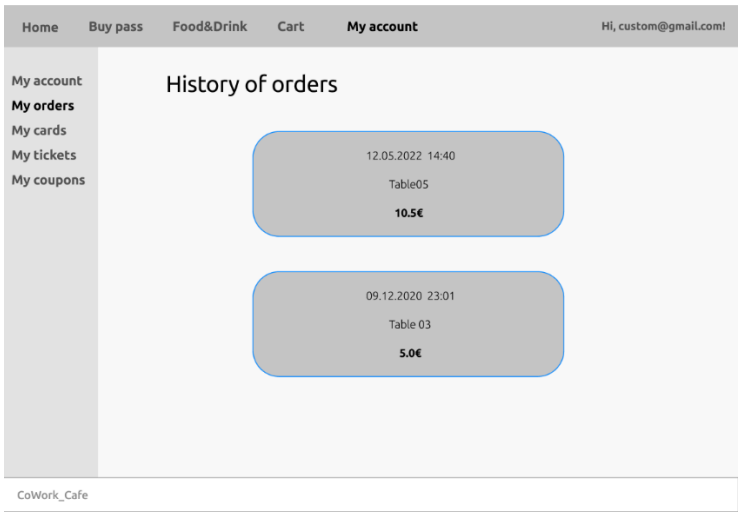
Ostukorvi vaates, mida saab näha joonisel 4, näeb klient kõiki menüüst lisatud kaupu ja ka valitud pileteid. Klient saab kaupade kogust muuta või need kustutada. Nimekirja lõpus näeb klient tellimuse hinda. Kasutajal on ka võimalus soodustuse saamiseks sisestada kupongi.

Nii piletite nimetusi ja hindu, menüü kategooriaid ja tooteid ning kupongide saamise tingimusi asutuse omanik või juht saab määrata ja muuta administraatori vaates.



Joonis 5. Tellimuse vormistamise vaade.

Nupp „*Order*“ ehk „Telli“ viib kliendi vormile, mis on kujundatud joonisel 5, kus on vaja kinnitada kohviku aadress, istumiskoht ja makseviis. Lehe lõppus on välja toodud tellimuse lõpplik hind koos rakendatud soodustusega juhul kui kupong oli lisatud. Vajutades nuppule „*Order*“ esitab kasutaja teelimuse.

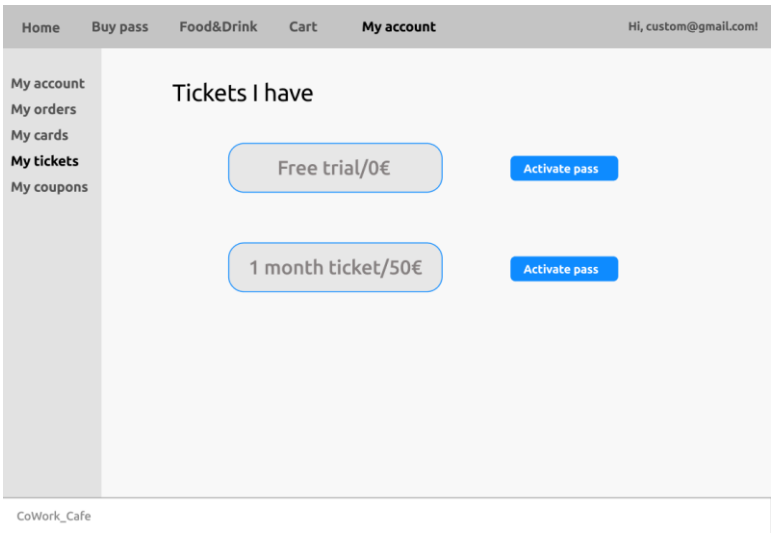


Joonis 6. Kasutaja tehtud tellimuste vaade.

„*My account*“ vaates, näeb klient mitmeid isikuandmete valikuid, nagu „*My account*“, „*My orders*“, „*My cards*“, „*My tickets*“, „*My coupons*“ ehk vastavalt „Minu konto“, „Minu tellimused“, „Minu kaardid“, „Minu piletid“, „Minu kupongid“. “*My account*” lehel näeb klient ees- ja perekonnanime, e-posti aadressi ja ka kliendi kategooriat: pronks, hõbe, kuld.

„*My orders*“ on leht, mis on kujutatud joonisel 6, mis näitab tehtud tellimuste nimekirja. Iga tellimuse juures kasutaja võib näha tellimuse aega, kohta, kuhu tellimus oli tehtud ning selle hinda.

„*My cards*“ vaates, klient võib näha kõiki tema poolt lisatud pangakaarte koos selle aegumise kuupäevaga, kustutada neid ära või lisada uusi kaarte.



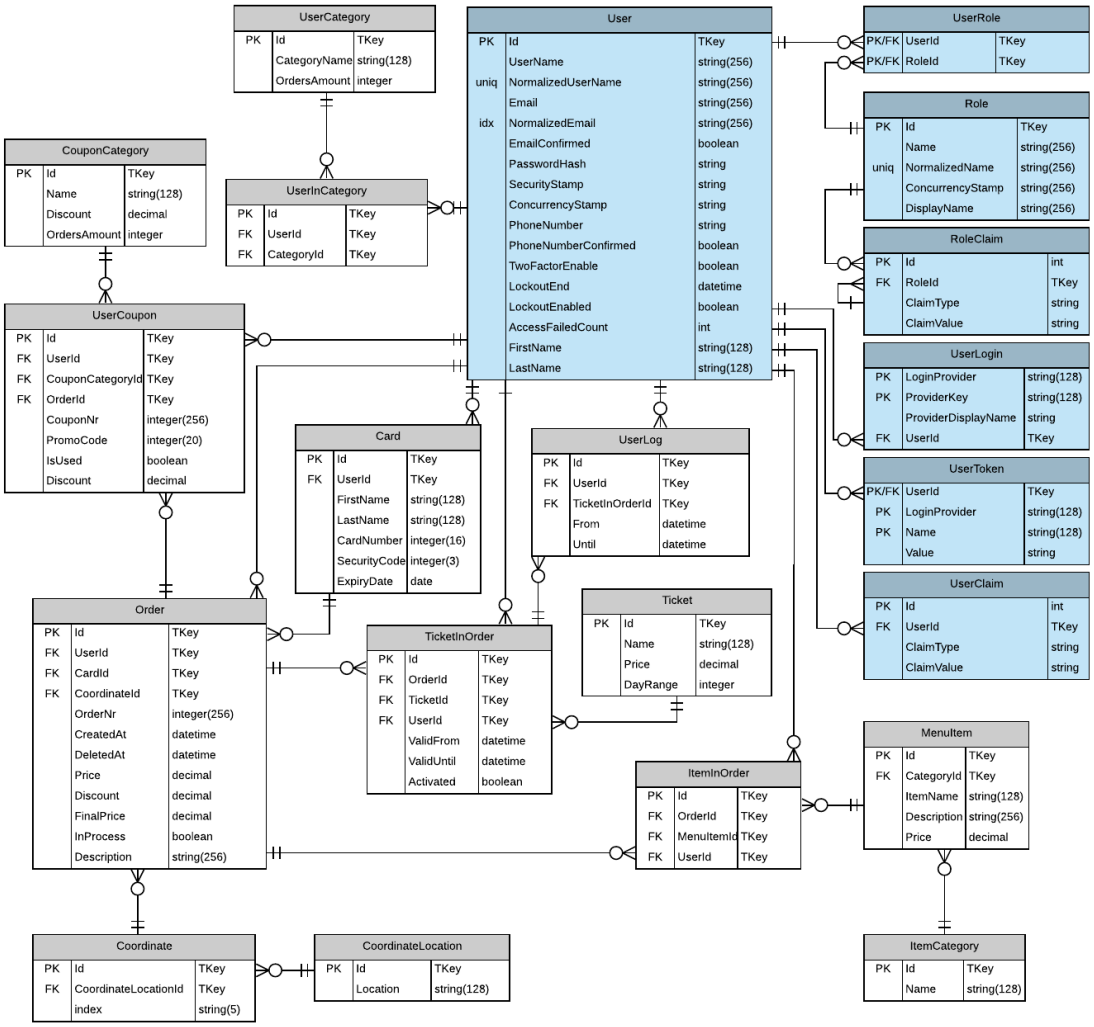
Joonis 7. Kasutajaga ostetud piletid.

„*My tickets*“ vaade, mis on kujutatud joonisel 7, näitab kliendi ostetud piletite loendit. Klient saab pileti aktiveerida, vajutades nuppu „*Activate pass*“ ehk „Aktiveeri sissepääs“. Kui klient aktiveerib pääsme, loob ta kirje andmebaasi, kus on teave selle kohta, kui kaua pilet kehtib ja klient saab kohvikut külastada. Need andmed on ka jaotises „*My tickets*“.

“*My coupons*” vaates kasutaja saab näha enda soodustusi ja kupongeid. Kupongid, mida klient saab tellimuste tegemisel kasutada põhinevad kliendikategoorial. Mida suurem on kategooria, seda rohkem kuponge on võimalik saada. Igal kupongil on soodustuskood, mida kasutaja võib sisestada tellimuse vormistamisel.

3.5 Andmebaasi olemi-suhte diagramm (*ERD*)

Eelnevas peatükkis tehtud kasutajaliidese prototüüp andis hea ülevaade nõuetest ning sellele tuginedes loodi andmebaasi disaini kirjeldatavat olemi-suhte diagramm ehk *ERD* (*Entity Relationship Diagram*). Diagrammi loomiseks kasutati *Licidchart* tööriista.



Joonis 8. Olemi-suhte diagramm (*ERD* mudel).

Joonisel 8 on välja toodud projekteeritud andmebaasi mudel. Sinise värviga märgitud tabelid kirjeldavad rakenduse kasutajate ja nende rollide hoidmise loogikat. Käesolevas töös kõiki kasutajatega seotud lahtreid ei kasutata, kuna luuakse rakenduse prototüübi ning esialgse versiooni, millele hiljem lisandub funktsionaalsus.

Ülejäänud tabelid aga võimaldavad luua ja täita eelmises peatükkis kirjeldatud funktsionalsust ja nõudeid.

3.6 Analüüsi kokkuvõte

Tehnoloogiate valiku analüüsis vaadeldi erinevaid võimalusi uue veebirakenduse loomiseks. Kuna rakendus koosneb tagarakendusest ja eesrakendusest, millele lisandub ka andmebaas selleks, et andmeid salvestada, valiti kaks raamistiku ning andmebaasimootori.

Andmebaasimoototriks valiti avatud lähtekoodiga *PostgreSQL*. Tagarakenduse loomiseks valiti laia ning võimsa funktsionaalsusega *.NET* raamistik. Eesrakenduse arendamiseks valiti *JavaScript* programmeerimiskeelel põhineva progressiivse ja lihtsa kasutamises *Vue.js* raamistiku. Loodavat rakendust ehitatakse järgides sibulaarhitektuuri mustrit, kuna see annab võimalust jagada rakenduse peaaegu sõltumatuteks osadeks ning eraldada andmebaasi loomise, selle kasutamise põhilist funktsionaalsust ning äriloogikad nii, et tulevikkus oleks lihtsam muuta ja täiendada rakendust.

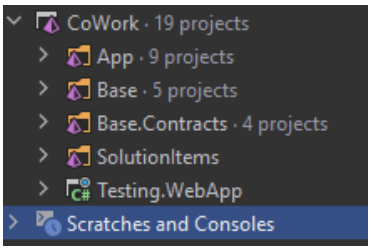
Loodava rakenduse funktsionaalsuse poolelt toodi välja kasutajalugusi, mille põhjal loodi visuaalne prototüüp ning andmebaasi olemi-suhte diagramm.

4 Veebirakenduse arendamine

Veebirakenduse arendamist jagati kolmeks suuremaks etapiks: tagarakenduse arendamine, eesrakenduse arendamine ning testimine. Järgmistes peatükkides vaadeldakse eelnevalt valitud tehnoloogiate abil rakenduse loomist, selle arhitektuuri ning failide ja kaustade loomise loogikat. Tuuakse välja koodi näiteid ning seletatakse taga-ja eesrakenduse omavahelise suhtlemise viisi.

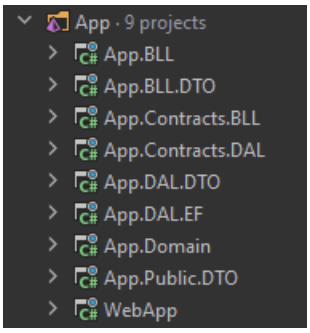
4.1 Tagarakendus

Tagarakenduse arendamiseks kasutatakse *C#* programmeerimiskeele *.NET* raamistiku. Rakenduse ehitamisel võetakse aluseks sibulaarhitektuuri põhimõtteid, mis on välja toodud peatukkis 3.3.3. Rakenduse kaustade paigaldust saab näha joonisel 9.



Joonis 9. Rakenduse põhilised kaustad.

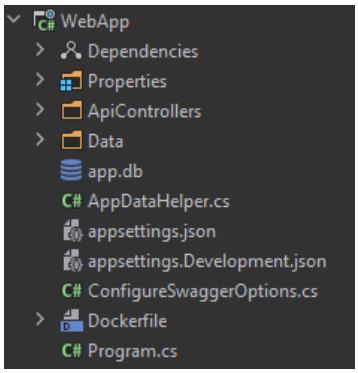
Kaustas „*App“* asub põhiline rakenduse funktsionaalsus koos sibulaarhitektuurilt tulenevate nelja kihiga, millega saab tutvuda joonisel 10. Kaustad nimega „*App.Domain*“ ning „*App.Public.DTO*“ täidavad domeenikihi funktsiooni. Seal saab leida olemite klasse, mille põhjal luuakse andmebaasi tabeleid *.NET Entity Framework* raamistiku abil, mis pakkub selleks lisafunktsionaalsust. Kaustades, mille nimedes esineb laiendus „*DAL*“ on *Repository Layer´i* ehk liidesekihi osad. Laiendusega „*BLL*“ on kaustad, mis on osa *Service Layer´*ist ehk teenuskihist.



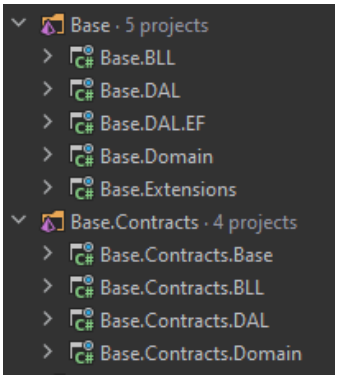
Joonis 10. „*App*“ kausta projektid.

Kaustas „*WebApp*“ asuvad klassid ja kaustad, mis kujundavad veebirakenduse ning täiendavad *Presentation Layer´it* ehk esitluskihti. Nendega saab tutvuda joonisel 11. Kaustas „*ApiControllers*“ asuvad *REST API* kontrollerid, mille abil tagarakenduselt saab küsida andmeid.

„*Program.cs*“ on põhiline konfiguratsioonide fail, kus seadistatakse erinevate paketide kasutamist, sealhulgas kasutaja autentimine ja autoriseerimine, *Swagger* teeki ehk projekti dokumentatsiooni loomise töörist, jne. Lisa seadistamine asub „*ConfigureSwaggerOptions.cs*“ failis.



Joonis 11. „*WebApp*“ kausta sisu.



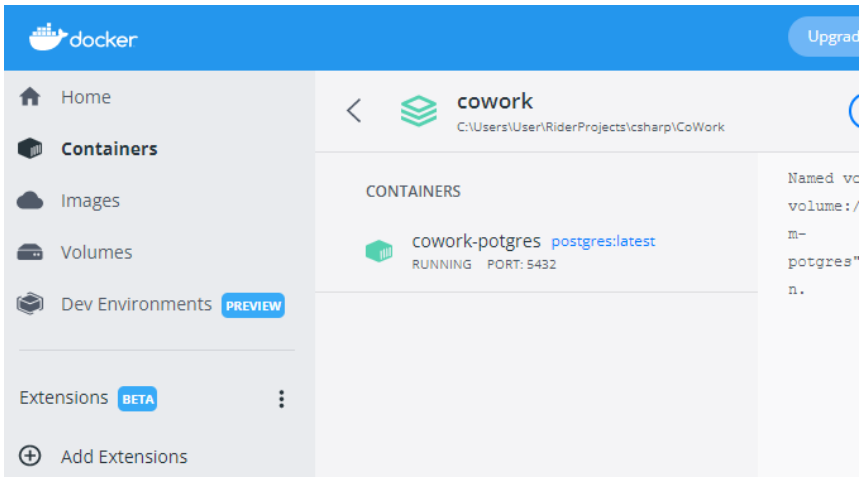
Joonis 12. „*Base*“ kaustade sisu.

Joonisel 12 on kujundatud laiendiga „*Base*“ kaustade sisu. Seal asuvad klassid, mis kirjeldavad erinevate kihide põhilist funktsionaalsust, mida kasutatakse „*App*“ kausta klassides. Selliste funktsionaalsuste hulka kuulub näiteks *Mapper´*i ehk teisaldusklassi loogika. Selleks, et ühe kihi *DTO*-st (*Data Transfer Object*-ist) ehk andmeedastusobjektist teha teise kihi *DTO*-d ning lõpus andmebaasi olemi objekti, kasutatakse *.NET* raamistiku „*Imapper*“ liidest, mis annab võimalust teisendada objekt õiget tüüpi objektiks.

Ülalnimetatud loogika klassid on eraldi väljatoodud kaustadesse selleks, et neid saaks taaskasutada, kuna ka teise rakenduse loomisel antud loogika ja käitumine ei muutu. Antud lähenemine aitab teha koodi taaskasutatavaks ning vähendada koodi mahu põhilises rakenduse kaustas, antud projekti puhul on selle nimeks „*App*“.

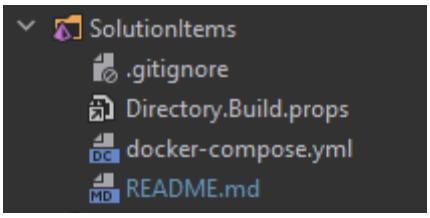
4.1.1 Andmebaasi kasutus

Analüüsi käigus valiti antud rakenduse andmete hoidmiseks *PostgreSQL* andmebaasimootorit. Andmebaasi konfigureerimist ning rakendamist otsustati teha läbi *Docker* tarkvara, mis jooksutab loodud andmebaasi konteinerina, mis töötab *GNU/Linux* baasil. Joonisel 13 on näidatud konteiner nimega „*cowork*“, mis on loodava rakenduse andmebaasi konteiner. *Docker* tarkvarast on mugav peatada ning uuesti tööle panna seda tarkvaraarenduse protsessi ajal.



Joonis 13. *Docker* andmebaasi konteiner.

Docker tarkvara seadistamiseks projekti failidest saab leida faili nimega „*docker-compose.yml*“, mis asub kaustas „*SolutionItems*“, mille asukoht on näidatud joonisel 14.



Joonis 14. *Docker* konfigureerimise fail.

Docker faili sisuga saab tutvuda joonisel 15. *Docker´*i konteineri loomisel aluseks võeti olemasolevat konteinerit nimega „*postgres:latest*“, mis juba sisaldab põhilisi seadistusi seega ei pea neid uuesti kirjutama.

```
version: '3.8'
services:
  exam-potgres:
    container_name: exam-potgres
    image: postgres:latest
    restart: unless-stopped
    environment:
      - POSTGRES_UER=postgres
      - POSTGRES_PASSWORD=postgres
    logging:
      options:
        max-size: 10m
        max-file: "3"
    ports:
      - "5433:5433"
    volumes:
      - exam-potgres-volume:/var/lib/postgresql/data
volumes:
  cowork-potgres-volume:
```

Joonis 15. *Docker* faili kood.

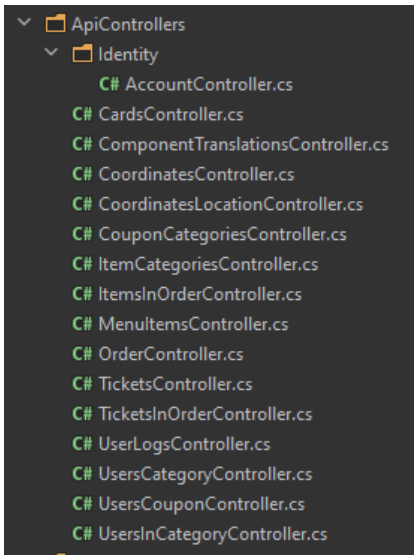
Joonisel 16 on kujutatud käsurea käsk andmebaasi migratsiooni käsk ning ka kustutamise käsk *Entity Framework* raamistiku abil. Esimene loob uue migratsiooni, mis sisaldab endas tabeleid genereeritud domeenikihi olemite klasside põhjal. Teine käsk käivitab andmebaasi migratsiooni kustutamise.

```
dotnet ef migrations add --project App.DAL.EF --startup-project WebApp --context AppDbContext Initial

dotnet ef database drop --project App.DAL.EF --startup-project WebApp --context AppDbContext
```

Joonis 16. Andmebaasi migratsiooni loomise ja kustutamise käsud.

4.1.2 Rakenduse kontrollerid

Tagarakendusega suhtlemiseks loodaval rakendusel on iga olemi jaoks tehtud kontrollerite klassid, nende loendiga saab tutvuda joonisel 17.



Joonis 17. Kontrollerite loend.

Kontrollerite tegemisel kasutati *Entity Framework* raamistiku võimalusi, mis teatud käsude terminalis käivitamisel, genereerib käsus määratud klassi põhjal kontrolleri koos põhiliste päringute meetoditega ehk *CRUD*. Ühe klassi kontrolleri genereerimise käsuga saab tutvuda joonisel 18.

```
dotnet aspnet-codegenerator controller -name CardsController
-m App.Domain.Card -actions -dc AppDbContext -outDir
ApiControllers -api --useAsyncActions -f
```

Joonis 18. Käsk kaardi kontrolleri genereerimiseks.

4.1.3 Dokumentatsioon

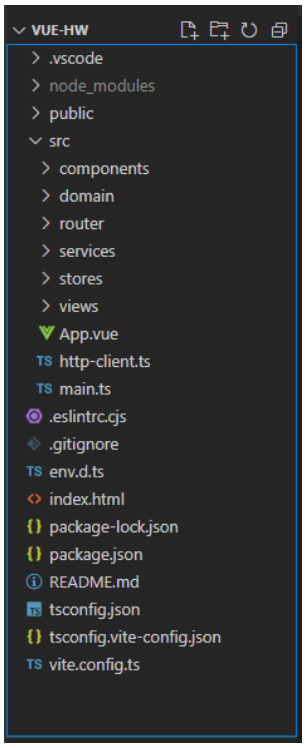
Vaatamata sellele, et antud töö raames valmib veebirakenduse prototüüp ning tulevikus rakenduse kasutusele võtmiseks on vaja seda edasi arendada, otsustati dokumenteerida ehk kommenteerida koodi. Tihtipeale mitte kõikidel ettevõtetel on reegliks kirjutada dokumentatsiooni, eriti väiksematel limiteeritud ressurside tõttu. Kui väike ettevõte kasvab suuremaks ning programm funktsionaalsus keerulisemaks, hakkatakse seda tegema. Dokumentatsioon on ettevõte edu võti [32].

Tagarakenduse *API* kontrollerite dokumenteerimiseks kasutatakse *.NET* raamistiku võimalusi. Iga kontrolleri meetodi ees on kommentari osa, mis seletab mida meetod võtab sisendiks ning mida väljestab ning annab üldist arusaamu mida saab antud meetodi välja kutsumisel saavutada ning mis on selle eesmärk.

4.2 Eesrakendus

Eesrakenduse on kasutajaliides, mis teeb päringuid tagarakenduse vastu. See võimaldab kasutaja jaoks andmete sisestamist ja kättesaamist lihtsustada. Analüüsi käigus valiti eesrakenduse tegemiseks *Vue.js* raamistik.

Projekti ülesehitusega saab tutvuda joonisel 19. Antud rakenduses kasutatakse *JavaScript´*i programmeerimiskeele laiendust *TypeScript*, mis teeb kohustuslikuks andmete tüüpide määramist. See aitab vältida koodis vigu ning teha koodi loetavamaks. *TypeScript´*i kasutamisega saab tutvuda joonisel 19 olevate failidega, mille laiend on „*.ts*“.



Joonis 19. Klientrakenduse ülesehitus.

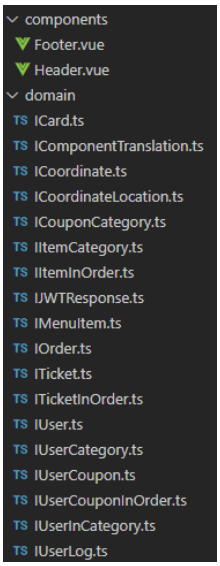
Failis „*http-client.ts*“ luuakse ühendust tagarakendusega (joonis 20). Ühenduse loomiseks määratakse link, kuhu päringuid saadetakse ning seadistatakse *header* ehk päis, mis kirjeldab saadetavate andmete tüüpi. Antud rakenduses kasutatakse *JSON* (*JavaScript Object Notation*) andmetevahenduse formaadi.

```
import axios from "axios";

export const httpClient = axios.create({
    baseURL: "https://localhost:7016/api/v1",
    headers: {
        "Content-type": "application/json"
    }
});

export default httpClient;
```

Joonis 20. Tagarakendusega ühenduse loomine.

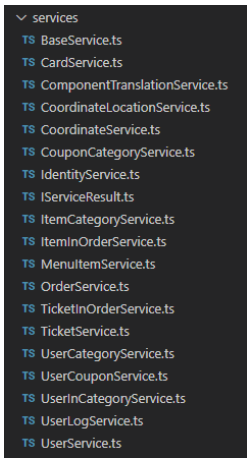
Joonisel 21 võib näha komponendide kausta, kus on kaks faili. Need failid vastutavad veebilehe *header´*i ja *footer´*i ehk päise ja jaluse eest. Need komponendid on välja toodud eraldi, kuna neid taaskasutatakse igal vaatel.

Samal joonisel saab tutvuda olemite kaustaga. Selles kaustas asuvad failid, kus on kirjeldatud rakenduse objektid. Need on sarnased tagarakenduses kirjeldatud objektidega, tagarakenduses objektidega toimub andmetöötlus, eesrakenduses kuvatakse objekte töödeldud andmetega välja. Objektid aitavad luua eesrakenduse loogikat, kuna andmetele ligipääs on lihtsam ja kiirem.



Joonis 21. Klientrakenduse komponendid ja olemid.

Klientrakenduses kasutatakse ka teenuse ehk „*service*“ kausta (joonis 22). Sarnaselt tagarakendusele, teenused on suuremas osa rakenduse äriloogika. Nendes klassides on meetodite ja funktsionaalsuse kirjeldus.



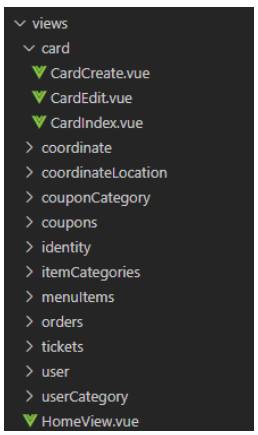
Joonis 22. Klintrakenduse teenusteklassid.

Joonisel 21 on välja toodud olemite klassid ning joonisel 22 on välja toodud teenusklassid, neid kaardistatakse kokku klassideks, mis asuvad kaustas „*stores*“. Antud klasside abiga on vaateid kiirem luua kuna on kiirem ligipääs meetoditele. Joonisel 23 saab näha, et ühe „*store*“ klassi loomisel importitakse sellese olemi ja teenuse fail.

```
import type { IMenuItem } from "@/domain/IMenuItem";
import { MenuItemService } from "@/services/MenuItemService";
```

Joonis 23. „*Store*“ kausta klassi loomise import.

Klientrakenduses on oluline ka kõikide nõutud funktsionaalsuste rakendamine, seega kaustas „*views*“ ehk vaated (joonis 24) on kirjeldatud iga kasutataud objekti jaoks kolm käitumise stsenaariumit. Näiteks kaustas „*card*“ ehk kaart, asuvad kolm faili mida kasutatakse pangakaardi lisamisel, muutmisel ning kustutamisel. Sarnane loogika on ka teiste olemite jaoks loodud.



Joonis 24. Klientrakenduse vaaded.

Joonisel 19 on kujutatud kaust „*router*“ ehk marsruuter, milles asub fail nimega „*index.html*“. Antud failis on kirjeldatud kogu eesrakenduse marsruutide kaardistus. Joonisel 25 on kujutatud vaateklasside importimist ning joonisel 26 imporditud vaadete seadistamist marsruutidega. Aluseks võetakse baasmarsruut ning sellele lisatakse laiendeid. Lisatud laiendi kaudu saab kasutaja ligipääsu vaadele, mis on sellele komponendina lisatud.

```
import CardIndex from "@/views/card/CardIndex.vue";
import CardCreate from "@/views/card/CardCreate.vue";
import CardEdit from "@/views/card/CardEdit.vue";
```

Joonis 25. Vaadete import.

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "home",
      component: HomeView,
    },
    {
      path: "/login",
      name: "login",
      component: Login,
    },
    {
      path: "/register",
      name: "register",
      component: Register,
    },
    { path: "/account", name: "account", component: UserIndex },

    { path: "/cards/create", name: "cardsCreate", component: CardCreate, props: true },
    { path: "/cards/edit/:id", name: "cardsEdit", component: CardEdit, props: true },
    { path: "/cards", name: "cards", component: CardIndex },
```

Joonis 26. Vaadete seadistamine linkidega.

Kõikide ülalloetletud vaated ning nende loogika kokkupanemine üheks veebileheks toimub failis „*App.vue*“ mis oli näha joonisel 19.

4.3 Testimine

Kuna valmiv rakendus on prototüüp siis põhirõhk langes rakenduse funktsionaalsuse testimisele. Selleks kasutati kaht testimise lähenemist: *Unit* ehk ühik ja *User Happy Path* ehk optimaalse kasutajakogemuse tee testimist. Lisaks sellele testiti ka eesrakenduse kasutajakogemust.

4.3.1 *Unit* testimine

Unit testid on testid, mis kontrollivad konkreetset väikest osa koodist ja selle funktsionaalsust. Nende testidega kontrollitakse võimalikult väike osa koodist nagu näiteks teatud meetodi testimine ja tänu sellele võib testide arv olla suur [33].

Antud rakenduses otsustati testida ühe teenusekihi klassi, kuna testimine võtab palju aega ning antud töö skoobis oleks kõikide klasside testimine liiga mahukas. *Unit* testid asuvad projektis eraldi kaustas „*Testing.WebApp*“, failis „*UnitTests.cs*“.

4.3.1 *User Happy Path* testimine

User Happy Path testimist kasutatakse rakenduse põhilise funktsionaalsuse näitamiseks ja kontrollimiseks. Valitakse oletatavad tegevused, mis suure tõenäosusega rakenduse kasutaja teeks ning hakatakse neid imiteerima. Valiti järgnev tegevuste järjekord:

- Registreeritakse uut kasutaja kontot.
- Kasutaja lisab ostukorvi ühe toote menüüst.
- Kasutaja lisab ostukorvi ühe pileti.
- Kasutaja lisab enda pangakaardi.
- Kasutaja vormistab tellimust.

Antud järjekord kirjeldab veebirakenduse põhilist eesmärki ning annab ka ülevaade selle toimimisest koodi poolelt. Uuel arendajal on projektiga lihtsam tutvuda, kuna saab näha milliseid meetodeid milleks kasutatakse. *User Happy Path* teste leidub kaustas „*Testing.WebApp*“ asuva „*IntegrationHappyFlowTest.cs*“ failis.

4.3.2 Kasutajakogemuse testimine

Eesrakenduse prototüübi testimiseks otsustati läbi viia väikse sihtgruppiga uuring tagasiside saamiseks. Tulevikkus peaks uuring olema tehtud suurema sihtgruppiga, et koguda väärtusliku statistikat rakenduse kasutamisest ning sellest teha järeldusi. Antud töö raames korraldatud uuring annab vaid üldist ülevaadet ning mõtteid kuidas võiks kasutajakogemust tulevikkus parandada.

Autor leidis neli vabatahtliku, kes erinesid oma vanuse, soo ja arvuti kasutamise oskustes. Järgmiselt on väljatoodud testitavate tutvustus ning nende kogemus teatud toidutellimise rakendustega.

1. Esimene testitav on 14. aastane noormees, kes kasutab arvutit igapäevaselt õppimiseks ning 3-5 korda nädalas mängib arvutimänge. Noormees ei ole kokkupuutunud toidutellimise rakendustega.

2. Teine testitav on 22. aastane noormees, kes kasutab igapäevaselt rakendusi toidu tellimiseks, teenuste broneerimiseks ja ostmiseks.

3. Kolmas testitav on 24. aastane noormees, kes kasutab rakendusi igapäevaselt, kuid lisaks sellele on tal kõrgharidus infotehnoloogia valdkonnas ning ta ise tegeleb veebirakenduste loomisega.

4. Neljas testitav on 53. aastane naine, kellel on kogemust rakendustega, kuid väga vähe.

Töö autor andis testitavatele rakenduse proovimiseks väikese tutvustuse. Tutvustust tehti selleks, et kasutaja saaks õigesti aru rakenduse kontekstist ning oskaks seda kasutada. Kasutajatel paluti ettekujutada, et nad on kohviku külastajad ning tahavad tellida enda lauale süüa. Samas paluti ka käia „*My account*“ vaades ning tutvuda rakenduse võimalustega. Pärast ülesande sooritamist, küsiti kasutajatelt tagasisidet.

Testitavalete meeldis rakendus ning see, et menüüst toidu saab valida kategooriate järgi. Lisaks meeldis neile idee, et ostetud pileti valideerimist saab läbi viia rakenduse kaudu. Positiivselt hinnati ka soodustuste süsteemi, kupoongid annavad motivatsiooni tulla järgmine kord tagasi ja tellida rohkem.

Kõik neli testitava tegid ühe ja sama märkuse uue veebirakenduse prototüübi kohta. Tellimuse vormistamisel, pärast kõikide väljade täitmist ning selle esitamist, suunatakse kasutajat veebilehe kogulehele ning see tekitab segadust, kuna kasutaja ei saa kindel olla, et kas tellimus oli ikka esitatud. Antud tähelepanek on kasutajapoolelt oluline, kuna kasutaja peaks igalt sammult saama programmilt tagasisidet, prototüübil antud funktsionaalsus puudub. Paranduse viiakse ellu järgmisel arendus etappil, mis jääb hetkesest skoobist välja.

5 Loodud veebirakenduse hinnang

Valmis on saanud veebirakenduse prototüüp, mis automatiseerib toitlustusasutuste ja ürituse korraldamise tööd. Rakenduse abil saab tellida toidu ning osta ja valideerida üritustele pileteid. Rakendus annab võimaluse ka soodustuste saamiseks. Loodud rakenduse prototüüp täidab töös väljatoodud nõudeid ja kasutajalugusid.

Tänu testimise läbiviimisele saadi veebirakendusele ka vajaliku tagasisidet, millest väljendub, et rakendus on kasutajasõbralik ning kasutaja saab kõikest vajalikust intuitiivselt aru.

Rakenduse projekti koodi saab leida peatükis Lisa 2. Veebirakenduse asutuse kliendi vaadetega saab tutvuda peatükis Lisa 3. Peatükis Lisa 4 on välja toodud ka administraatori vaated.

5.1 Edasiarendus

Töö käigus toodi välja kasutajalood ja nõuded, mida arendatakse antud skoobist väljaspool (peatükk 3.2) ning testimisel saadud tagasiside andis ideid kuidas saaks rakendust tulevikus täiendada.

Veebirakenduse prototüüp hästi ilustreerib tulevase rakenduse käitumist, ennem kui seda saab kasutusele võtta ning pakkuda erinevatele asutustele on vajalik korraldada veel arendus etappe.

Peale täiendavate etappide korraldamist ning lahenduste testimist suuremate sihtgruppidega on plaan koostada äriplaan kus tuleks analüüsida rakenduse kasu teatud asutuse näitel ja kas tööjõu vajadus langeb programmi integreerimisega. Peale analüüsimist tuleks äriplaani esitlemine kliendile ehk selle valdkonnaga tegelevale asutusele, kes on selle programmi sihtgrupp.

Lisaks toitlustusasutustele, rakendus aitaks ka korraldada festivale, kus on vajalik piletite ostmise süsteem. Ka festivalil olevaid toitupakkujaid võib ühendada uue süsteemiga ning lahendada pikkade järjekordade probleemi. Tulevikus plaanitakse ka luua kõikide partnerite jaoks ühise platvormi, mis ühendaks kõiki asutusi ühes kohas, mis aitab ka väiksematel asutustel jõuda laiema sihtgruppini.

6 Kokkuvõte

Töö käigus loodi veebiakenduse prototüüp, mis aitab automatiseerida toitlustust ning kogutud andmetega parandada tööd. See lahendus aitab korraldada toidupakkuvatel asutustel üritusi, müüa pileteid, eelnevalt broneerida asutuse laudu, tellida toidu broneeritud lauale ning valideerida ostetud pileteid.

Töö analüütilises osas anti ülevaade olemasolevatest lahendustest ning nende tugevamatest ja nõrgematest küljedest, mille põhjal koostati uuele rakendusele nõudeid. Analüüsiti sobivaid tehnoloogiaid, programmeerimiskeeli ja raamistike taga- ja eesrakenduse loomiseks, mille põhjal valiti sobivaimad. Anti ülevaade loodud veebirakenduse arhitektuurist ning seletati arendamise etapid lahti. Valminud veebirakenduse prototüübile viidi läbi testimine ning saadi kasutajatelt tagasisidet.

Töö käigus kõik püstitatud nõuded said täidetud ning testimise käigus saadud tagasisided on positiivsed. Veebirakendus on valmis kasutamiseks.

Kasutatud kirjandus

[1] Tõnu Väät. (2021, oktoober) E-poest on toitu tellinud pea 70% tallinlasi ning peamine on isiklik soovitus. [Online] https://www.e-kaubanduseliit.ee/uudised/e-poest-on-toitu-tellinud-pea-70-tallinlasi-ning-peamine-on-isiklik-soovitus

[2]Sandra Lepik. (2020, august) UURING: Üle poolte peavad toidu koju tellimist mugavamaks kui kokkamist. [Online] https://pealinn.ee/2020/08/03/uuring-ule-poolte-peavad-toidu-koju-tellimist-mugavamaks-kui-kokkamist/

[3] Bolt. (2023, veebruar) About Bolt [Online] https://bolt.eu/en/careers/about-bolt/

[4] Ylle Tampere. (2019, august) Tallinna kesklinnas saab tellida toitu nüüdsest ka Bolt Food rakenduse kaudu. [Online] https://www.accelerista.com/uudis/eesti/bolt-food-tallinnas/

[5] Bolt. (2023, veebruar) Bolt Food. [Online] https://partners.food.bolt.eu/et-ee/

[6] Fienta. (2023, veebruar) Korraldajale. [Online] https://fienta.com/et/korraldajale

[7] Fienta. (2023, veebruar) Võimalused. [Online] https://fienta.com/et/voimalused

[8] Collins Dictionary. (2023, veebruar) English Dictionary. [Online] https://www.collinsdictionary.com/dictionary/english/catering

[9] AB Catering. (2023, veebruar) AB CATERING – TELLI TOITLUSTUS. [Online] https://abcatering.ee/?gclid=Cj0KCQiArsefBhCbARIsAP98hXTKa6sgi4dzSpINM7EM1k1eyUN9WHQ7AJx8vTLBRQyuXMtm6DWrQLEaAgt2EALw_wcB

[10] Andrew Gromenko. (2022, september) What Is The Difference Between A Programming Language And A Framework? [Online] https://code-care.com/blog/what-is-the-difference-between-a-programming-language-and-a-framework

[11] Oracle. (2023, märts) What Is a Database? [Online] https://www.oracle.com/cis/database/what-is-database/

[12] Oracle. (2023, märts) What Is a Relational Database (RDBMS)? [Online] https://www.oracle.com/database/what-is-a-relational-database/

[13] Mahesh Chand. (2023, märts) What are the Most Popular Relational Databases (2023). [Online] https://www.c-sharpcorner.com/article/what-are-the-most-popular-relational-databases/

[14] Amazon. (2023, märts) What Is PostgreSQL? [Online] https://aws.amazon.com/rds/postgresql/what-is-postgresql/

[15] Oracle. (2023, märts) Oracle MySQL. [Online] https://shop.oracle.com/apex/product?p1=MySQL

[16] Shahzeb Ahmed. (2022, november) MariaDB vs MySQL: A Detailed Comparison. [Online] https://www.cloudways.com/blog/mariadb-vs-mysql/

[17] SQLite. (2023, märts) Appropriate Uses For SQLite. [Online] https://www.sqlite.org/whentouse.html

[18] Liz Simmons. (2023, märts) Front-End vs. Back-End: What's the Difference? [Online] https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/

[19] Simplilearn. (2023, veebruar) What Is Spring Framework and Its Advantages. [Online] https://www.simplilearn.com/tutorials/spring-boot-tutorial/what-is-spring-framework-and-its-advantages

[20] Javatpoint. (2023, märts) Spring Boot Tutorial. [Online] https://www.javatpoint.com/spring-boot-tutorial

[21] InterviewBit. (2022, november) .NET Core vs .NET Framework. [Online] https://www.interviewbit.com/blog/net-core-vs-net-framework/

[22] Microsoft. (2023, märts) What Is .NET? [Online] https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[23] Tycoonstory. (2022, veebruar) The Advantages and Disadvantages Of .NET Framework Programming. [Online] https://www.tycoonstory.com/technology/the-advantages-and-disadvantages-of-net-framework-programming/

[24] Django. (2023, märts) Meet Django. [Online] https://www.djangoproject.com/

[25] Oleg Kopachovets. (2022, oktoober) Express JS vs Node JS: Why it's Time to Migrate? [Online] https://procoders.tech/blog/express-js-vs-node-js/

[26] Sardar Mudassar Ali Khan. (2022, juuli) Onion Architecture In ASP.NET Core 6 Web API. [Online] https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-6-web-api/

[27] ypsjnv2013. (2023, märts) Difference between HTML and CSS. [Online] https://www.geeksforgeeks.org/difference-between-html-and-css/

[28] Godwin Alexander Ekainu. (2023, jaanuar) 9 Best JavaScript Frameworks to Use in 2023. [Online] https://ninetailed.io/blog/best-javascript-frameworks/

[29] React. (2023, märts) React. [Online] https://reactjs.org/

[30] Vue.js. (2023, märts) Introduction. [Online] https://vuejs.org/guide/introduction.html

[31] Figma. (2023, märts) Products. [Online] https://www.figma.com/prototyping/

[32] Pierre Bourque, Richard E. (Dick) Fairley (2013) SWEBOK V3, Guide to the Software Engineering Body of Knowledge. [Online] https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf

[33] SMARTBEAR. (2023, aprill) What Is Unit Testing? [Online] https://smartbear.com/learn/automated-testing/what-is-unit-testing/

[34] Bootstrap. (2023, aprill) Bootstrap. [Online] https://getbootstrap.com/

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

Mina, Viktoria Mihhailova

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Toitlustusasutuste klienditeeninduse automatiseerimise prototüüp", 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.

13.05.2023

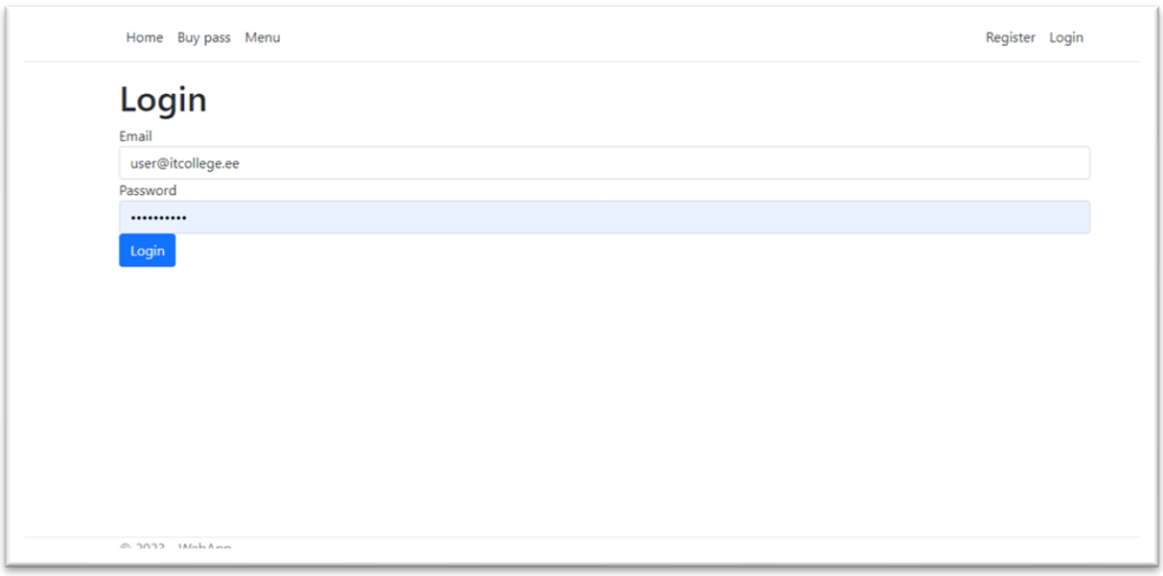
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või 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.

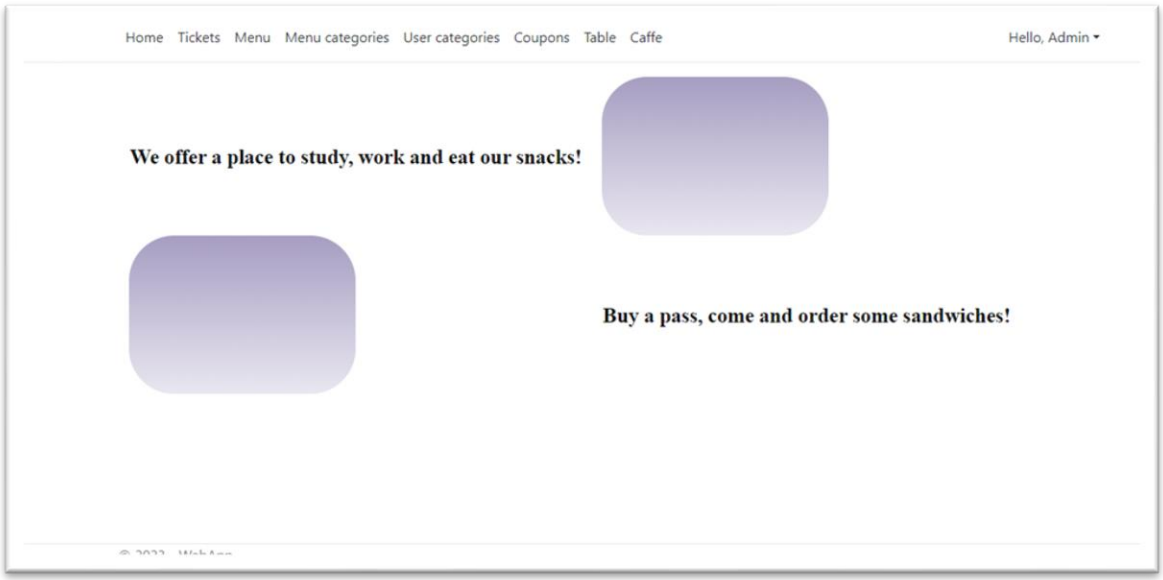
Lisa 2 – Ees- ja tagarakenduse versioonihaldus

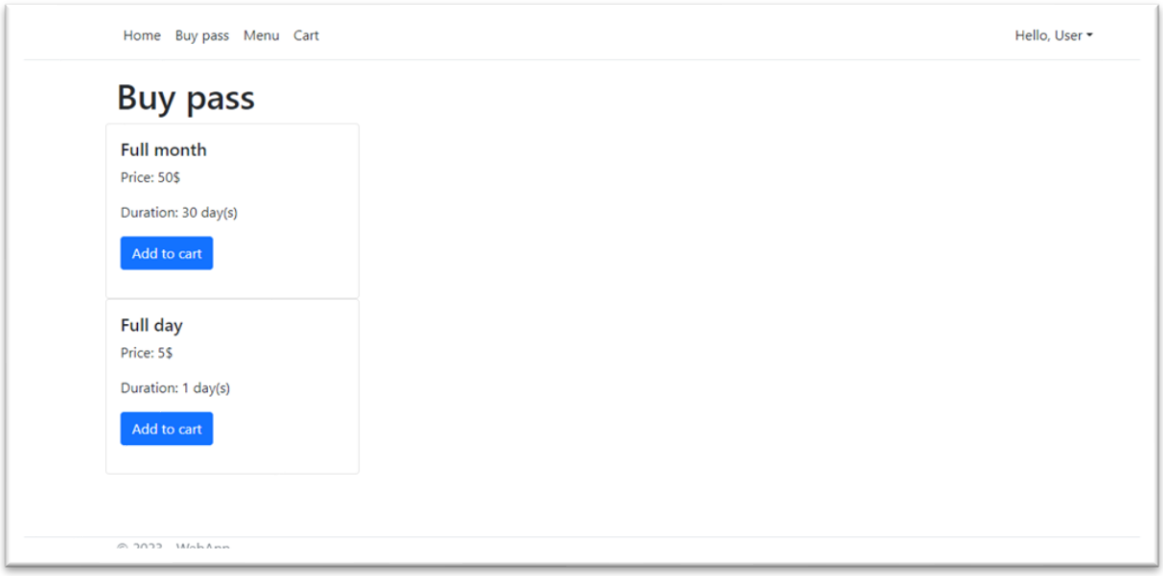
Eesrakendus: https://github.com/vikmihh/Projects/tree/main/client/vue

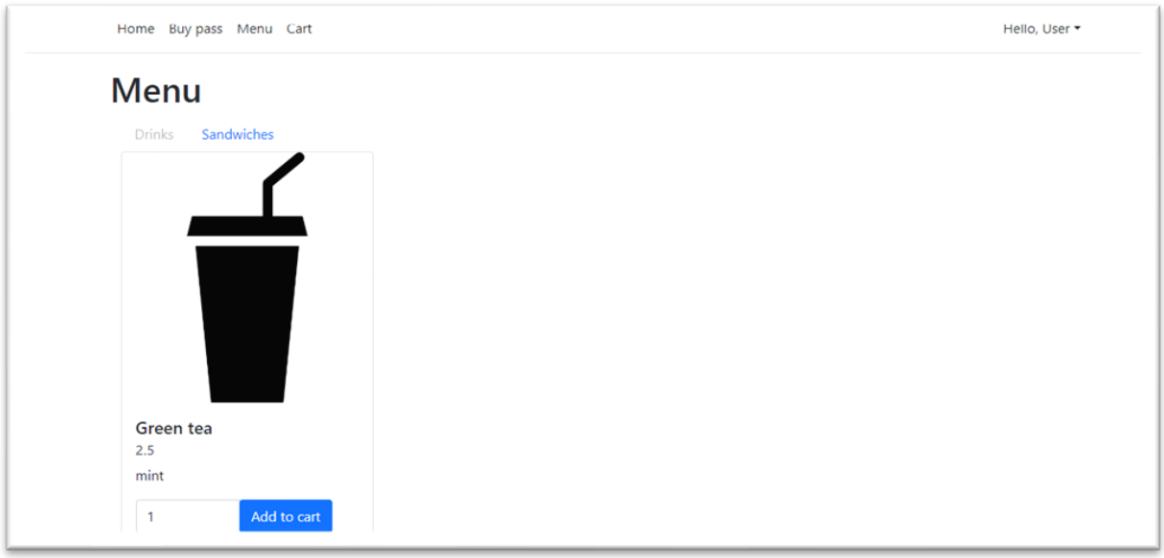
Tagarakendus: https://github.com/vikmihh/Projects/tree/main/server/CoWork

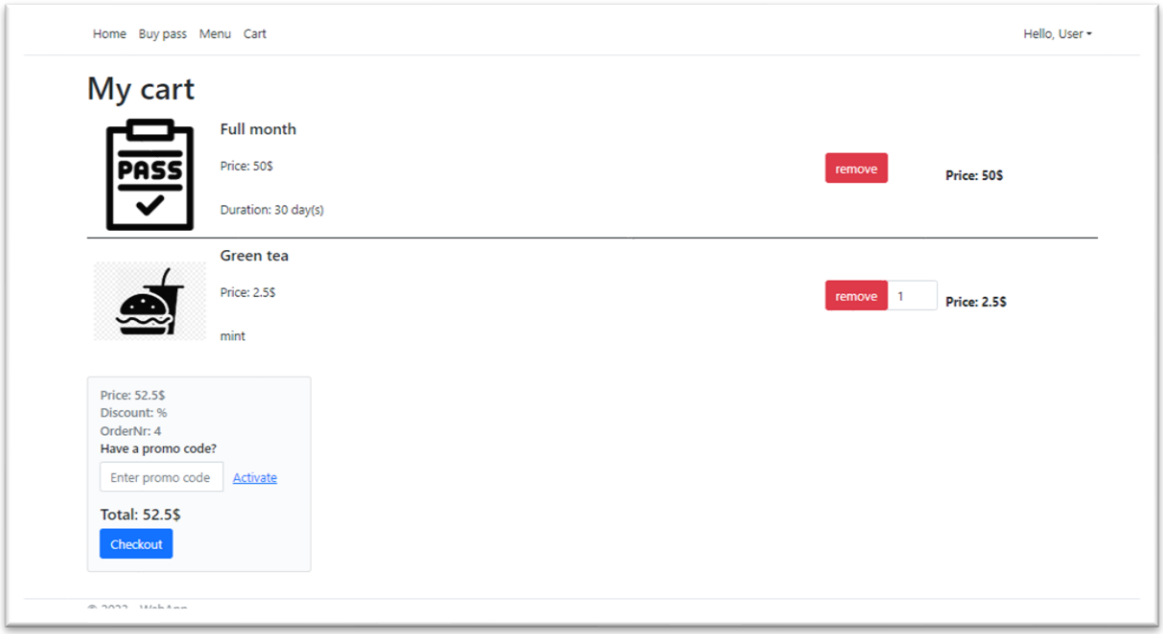
Lisa 3 – Veebirakenduse vaated

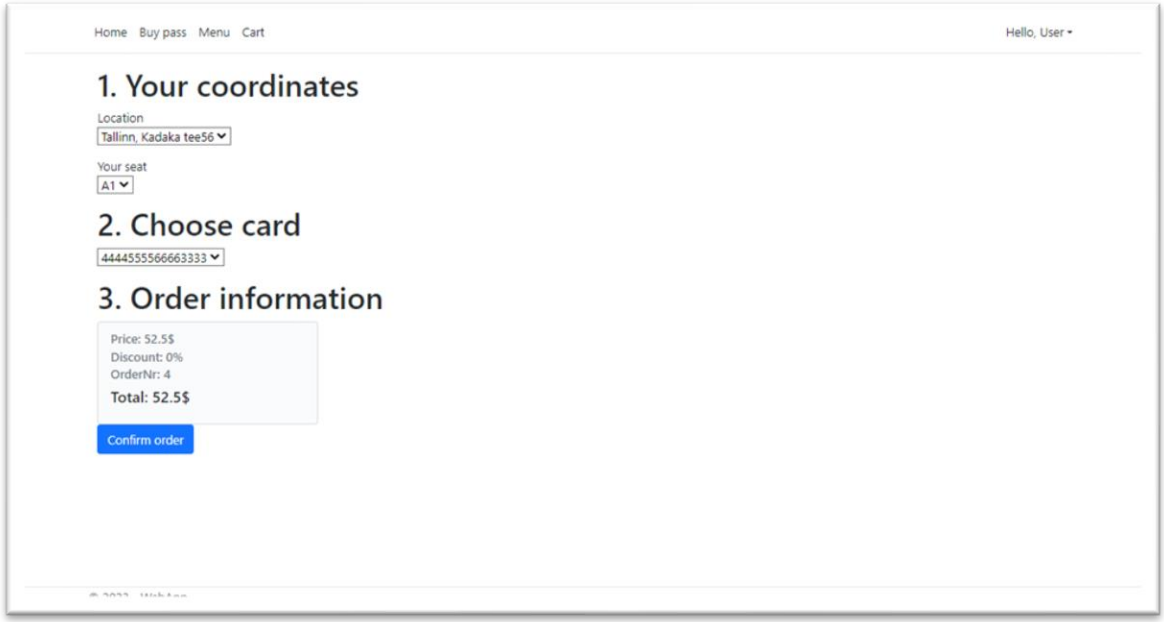




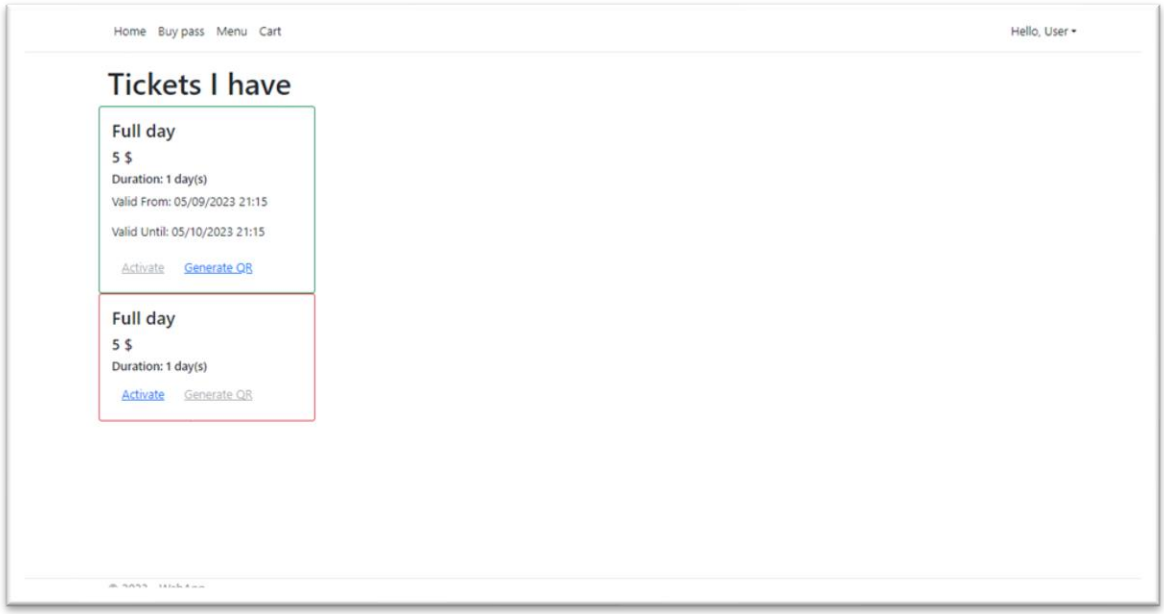

Home Buy pass Menu Cart
Hello, User
Buy pass
Full month
Price: 50$
Duration: 30 day(s)
Add to cart
Full day
Price: 5$
Duration: 1 day(s)
Add to cart


Home Buy pass Menu Cart
Hello, User
Menu
Drinks Sandwiches
Green tea
2.5
mint
1
Add to cart

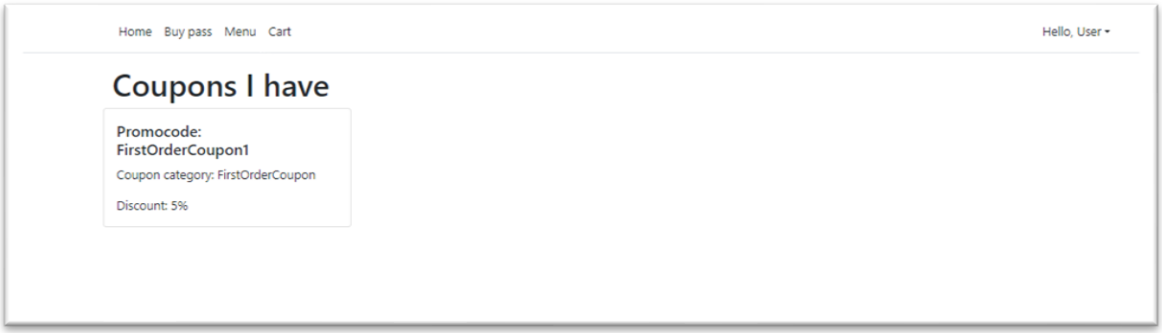

Home Buy pass Menu Cart
Hello, User
My cart
Full month
Price: 50$
Duration: 30 day(s)
remove
Price: 50$
Green tea
Price: 2.5$
mint
remove
1
Price: 2.5$
Price: 52.5$
Discount: %
OrderNr: 4
Have a promo code?
Enter promo code
Activate
Total: 52.5$
Checkout


Home Buy pass Menu Cart
Hello, User
1. Your coordinates
Location
Tallinn, Kadaka tee56
Your seat
A1
2. Choose card
4444555566663333
3. Order information
Price: 52.5$
Discount: 0%
OrderNr: 4
Total: 52.5$
Confirm order


Home Buy pass Menu Cart
Hello, User
My account
Orders
Cards
Tickets
Coupons
Logout
Orders history
OrderNr: 1
Price: 5$
Discount: 0 %
Final price: 5$
CardNr: 4444555566663333
Ordered At: 06/01/2022 14:39
OrderNr: 2
Price: 8$
Discount: 0 %
Final price: 8$
CardNr: 4444555566663333
Ordered At: 06/01/2022 14:55
OrderNr: 3
Price: 7.5$
Discount: 0 %


Home Buy pass Menu Cart
Hello, User
Tickets I have
Full day
5 $
Duration: 1 day(s)
Valid From: 05/09/2023 21:15
Valid Until: 05/10/2023 21:15
Activate
Generate QR
Full day
5 $
Duration: 1 day(s)
Activate
Generate QR


Home Buy pass Menu Cart
Hello, User
Activated
Valid from: 05/09/2023 23:42
Valid until: 05/09/2023 23:43
Back to List


Home Buy pass Menu Cart
Hello, User
Coupons I have
Promocode:
FirstOrderCoupon1
Coupon category: FirstOrderCoupon
Discount: 5%

Lisa 4 – Veebirakenduse administraatori vaated