TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ilja Ivanov 193651IADB

Restorani toidu tellimise mobiilirakenduse ja tagarakenduse arendamine

Bakalaureusetöö

Juhendaja: German Mumma
MSc
Äriinfotehnoloogia

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: Ilja Ivanov

Annotatsioon

Käesolev lõputöö kirjeldab mobiili- ja tagarakenduse arendamisprotsessi restorani jaoks.

Esimeses osas kirjeldab autor põhjalikumalt töö tausta ja probleemi.
Teises osas analüüsitakse tellija nõudeid, tagarakenduse ja andmebaasi arhitektuuri, antakse ülevaade olemasolevatest rakendustest ja võrreldakse loodud rakendust teiste samas valdkonnas toimivate teenusepakkujate rakendustega. Lisaks tutvustatakse rakenduse arendamisel kasutatud tehnoloogiaid. Kolmas osas kirjeldatakse tagarakenduse ja mobiilirakenduse struktuuri. Neljas osas tutvustatakse mobiilirakenduse funktsionaalsust, mida illustreeritakse ekraanitõmmistega rakendusest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 6 peatükki, 22 joonist ja 1 tabelit.

Abstract

Development of Restaurant Food Ordering Mobile and Backend Application

This thesis describes the development process of a food ordering mobile application and its backend.

In the first part, the author describes the background and the problem on which the work is based in more detail. The second part analyzes the client's requirements, the application backend and the architecture of the database. Additionally, an overview of select applications of competitors in similar field is given, and a comparison is made with those softwares. After that, the author introduces the technologies used in the development of their application. The third part describes the structure of the application’s backend and the mobile application. The fourth part gives an overview of the functionality of the mobile application, which is illustrated with screenshots from the application.

The thesis is written in Estonian and contains 26 pages of text, in 6 chapters, contains 22 figures, and 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* - rakendusliides
CRUD	*Create, read, update, and delete* - püsiva salvestamise neli põhitoimingut
HMAC	*Hash based Message Authentication Code* - sõnumi autentimiskoodi tüüp, mis saadakse autentitavatel andmetel krüptograafilise räsifunktsiooni täitmisel ja salajasel jagatud võtmel
HTTP	*Hyper Text Transfer Protocol* - protokoll teabe edastamiseks arvutivõrkudes
JSON	*JavaScript Object Notation* - formaat andmete salvestamiseks ja transportimiseks
MVCC	*Multiversion Concurrency Control* - on samaaegsuse kontrolli meetod, mida tavaliselt kasutavad andmebaasihaldussüsteemid, et pakkuda samaaegset juurdepääsu andmebaasile
ORM	*Object-Relational Mapping* - programmeerimistehnika, mille puhul kasutatakse metaandmete deskriptorit objektikoodi ühendamiseks relatsiooniandmebaasiga
Prisma	objektide relatsioonikaardistaja (ORM), mis kasutab kohandatud skeemi määratlemise keelt (SDL), mis kirjutab automaatselt migratsioonid ja genereerib tüübikindla koodi
REST	*Representational State Transfer* – veebiteenusega suhtlemise liidese arhitektuur
SOLID	Projekteerimispõhimõtete kogum, mida kasutatakse objekt-orienteeritud tarkvaraarenduses

Sisukord

Tabelite loetelu

1 Sissejuhatus

Käesoleva lõputöö raames arendati hübriid-mobiilirakendus ja tagarakendus sushirestorani toidu tellimise jaoks.

Projekti arendamine käis koostöös tellijaga ja teise lõpetajaga. Valmis projekt koosneb kolmest osast: veebirakendus, mobiilirakendus ja ühine tagarakendus. Tagarakenduse vastutusalad olid jagatud nii:

- autori ülesandeks oli arendada mobiilirakenduse ja tagarakenduse osad, mis vastutavad turvalisuse, autentimise ja tellimisloogika eest
- teise lõpetaja ülesandeks oli arendada veebirakendust ja tagarakenduse osa, mis vastutab peamiselt CRUD-i toimingute eest

Lõputöö põhieesmärgiks oli saada valmis töötav mobiilirakendus, mis võimaldab klientidel tellida toitu koju ning restorani omanikel hallata toitude menüüd ja sooduspakkumisi ning tellimuse staatust. Valminud mobiilirakenduse funktsionaalsust võrreldakse käesolevas töös juba olemasolevate sarnaste rakendustega ning antakse hinnang töö raames valminud mobiilirakenduse funktsionaalsuse kvaliteedile.

Tuginedes ülevaatele olemasolevatest rakendustest, sätestasid tellija ja autorid enne mobiilirakenduse arendamist rakendusele järgmised nõuded:

- rakendust saab kasutada ainult registreeritud kasutaja
- peab toimuma registreerimine ja autoriseerimine
- võimalus vaadata toodete loendit
- võimalus filtreerida toitu kategooria järgi
- võimalus vaadata toidu koostise infot
- võimalus lisada toitu lemmikutesse
- võimalus lisada toitu ostukorvi
- võimalus esitada tellimust
- võimalus oma tellimuste ajalugu vaadata
- võimalus kontrollida tellimuse staatust

- võimalus vaadata kehtivaid sooduspakkumisi
- võimalus maksta kaardiga

Lõputöö kirjalik osa koosneb sissejuhatusest, neljast peatükist ja kokkuvõttest.
Esimeses peatükis kirjeldab autor põhjalikumalt töö tausta ja probleemi.
Teises peatükis analüüsitakse tellija nõuet, tagarakenduse arhitektuuri ja selle loogikat, andmebaasi arhitektuuri, olemasolevaid rakendusi ja võrreldakse loodud rakenduse teiste rakendustega, tutvustatakse rakenduse arendamisel kasutatud tehnoloogiaid. Neljas peatükis kirjeldatakse tagarakenduse ja mobiilirakenduse struktuuri. Viies peatükis kirjeldatakse mobiilirakenduse funktsionaalsust, mida illustreeritakse ekraanitõmmistega rakendusest.

2 Taust

COVID-19 pandeemia on mõjutanud inimeste suhtumist toidu tellimise rakendustesse. Pandeemia piirangute tõttu olid toitlustusettevõtted sunnitud pakkuma toidu kontaktivaba kojuvedu. Teatud uuringud näitavad, et tõenäoliselt jätkavad inimesed ka peale pandeemia lõppu toidu tellimise rakenduste kasutamist, kuna pandeemia aitas kaasa rakenduste usaldusväärsuse tõstmisele, nende arengule ja funktsionaalsuse pidevale parendamisele [1].

2.1 Probleem

Toitlustusettevõttel on klientidele kojuveoteenuse pakkumisel võimalik kasutada vahendajate rakendusi, mis koondavad oma platvormil mitmeid restorane, või luua enda rakendus. Mõlemal suunal on omad plussid ja miinused.

Vahendaja teenuse kasutamisel on mitu eelist: vahendaja ise tegeleb rakenduse arendamisega ja kullerite töö haldamisega. Suured toidutellimise platvormid on turul hästi reklaamitud ja nähtavad, seega on nende kaudu lihtsam kliendini jõuda. Suurimaks miinuseks on aga kõrged vahendustasud, vt. nt Traynor et al (2022) uuringut USA toidutellimise vahendajatest [2]. Eesti toiduettevõtted on samuti selle probleemiga kokku puutunud ning otsinud alternatiive vahendajatele nagu Bolt ja Wolt, kes küsivad keskmiselt 25-30 protsenti vahendustasu iga tellimuse pealt [3]–[4]. Samuti saab toiduettevõtte müüa vahendajate platvormidel ainult piiratud valikut oma toodetest. Peetri Pitsa juhi sõnul on murekohaks ka see, et vahendajate teenuseid kasutades kaob toitlustusettevõttel kontakt oma kliendiga, sest koostööpartner haldab kõik ise, ja müüjale pole teada, millised on kliendi ootused [4].

Oma toidutellimuse rakenduse loomine ja arendamine on aga mahukas ja kallis. Samas võib see investeering tasuda ära, kuna siis on võimalik toitlustusettevõttel pakkuda oma kliendile paremat ja talle suunatud teenust ning teenida suuremat kasumit kohale toimetatud toidust [4].

Mitmed toitlustusettevõtted, eriti just suuremad neist, on huvitatud oma mobiilirakenduse loomisest, kuna see võimaldab luua otsest kontakti oma kliendiga, vältida vahendaja

kõrgeid teenustasusid ja muid tõrkeid. Peamiseks takistuseks selle juures on konkurents, nimelt on toidutellimise vahendajate rakendustel on juba väljakujunenud suured kasutajaskonnad, kes eelistavad suure restoranide valikuga toidutellimise platvorme ja võivad vältida üksikute restoranide rakenduste alla tõmbamist oma nutiseadmesse.

Seega ühele restoranile arendatav eraldi rakendus peab olema konkurentsivõimeline, st tagama sama head kasutajakogemust nagu vahendajate mobiilirakendused ning olema funktsionaalsuse osas kvaliteetne.

2.2 Veebirakendus vs. mobiilirakendus

Tellija soovis luua enda ettevõttele nii veebirakenduse kui mobiilirakenduse. Veebirakenduse lõi ettevõttele M. Kostjaev oma bakalaureusetöö raames "Veebirakenduse arendus restoranist toidu tellimiseks" [5]. Veebirakendus pidi ühtlasi toimima ettevõtet tutvustava veebilehena.
Mobiilirakenduste eeliseks on see, et enamikel juhtudel need on kiiremini kui tavalised veebirakendused ja need saavad töötada taustal ning saata automaatseid teavitusi. Veebilehitsejas on näiteks tellimuse staatust ebamugavam jälgida, sest selle jaoks peab brauser olema kogu aeg avatud. Ja viimaseks, mobiilirakendused on loodud spetsiaalselt nutiseadmes kasutamiseks, samal ajal kui veebirakenduse UI/UX ja funktsionaalsus on mõneti piiratud veebibrauseriga.

3 Analüüs

3.1 Tellija nõuete analüüs

Selleks, et arendada konkurentsivõimeline rakendus, tuleb arvestada teguritega, mis toetavad kliendi motivatsiooni rakendust kasutada. Nende hulgas on nii kasutajakogemuse disainimine kui erinevate kasutaja kaasamise mehhanismide lisamine. Erinevates uuringutes (Chan et al, 2014; Ray et al, 2019; Tandon et al 2021) tuuakse välja selliseid hästi töötavaid kaasamismeetodeid:

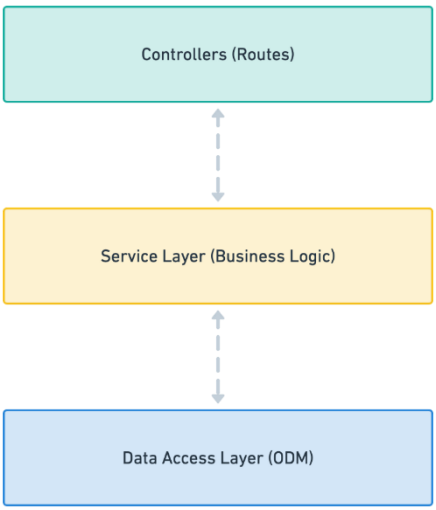
- kupongid, allahindlused (alalised ja ajutised), lojaalsusprogrammid, soovitusprogrammid ja reklaamid
- lihtne tellimuse koostamise ja esitamise protsess, sealhulgas erinevad filtrid ja võimalus kontrollida tellimuse staatust
- teenuse autentsus - toitlustamisettevõtte omapärasuse esiletoomine, sealhulgas info töötajate kohta ja taustalood kõige populaarsemate toitude kohta

Tellija nõuete põhjal pandi kirja kasutajalood (Lisa 4), tekitati äriprotsesside voog (Lisa 3) ja kasutusjuhtude diagramm (Lisa 5).

3.2 Tagarakenduse arhitektuuri analüüs

Tagarakenduse arhitektuur põhineb kolmekihilisel arhitektuuril (inglise keeles *3-Layers architecture*), mis jagunevad järgmiselt:

1. Kontrollerid (Controllers) – kihi eesmärk on vastu võtta eesrakenduse päringud ja määrata API teed
2. Teenuste kiht (Service Layer) - sisaldab ainult äriloogikat. Näiteks tehakse siin kõik CRUD-i toimingud ja meetodid andmete loomise, salvestamise ja värskendamise viiside määramiseks
3. Andmejuurdepääsu kiht (Data Access Layer) - see kiht määrab andmetele ligi pääsemise ja andmebaasis tehtavate toimingute viisid



Joonis 1. 3-Kihilise arhitektuuri diagramm

Antud jagunemine kihtideks võimaldab jagada rakendust paremini hallatavateks üksusteks ja tagab teist implementeerimist. Igal kihil on oma vastutusala, kas siis liides, äritöötlus või päringute esitamine. See on efektiivsem kogu koodi ühte kohta koondamisest.
Kuna kihid on üksteisest eraldatud, lihtsustab see muudatuste tegemist. Neid saab teha ühe kihi piires või kaasata ainult lähimat kihti, ilma kogu programmi mõjutamata [8].

Nest.js tagab CRUD-operatsioonid, kasutades kontrollereid. Kui kasutaja lisab mingi toote oma ostukorvi, siis iga lisamise puhul tehakse POST-päring serverile, kus seotakse kasutaja korv tootega. Alternatiivselt oleks võimalik salvestada toode spetsiaalses kontekstis, mida pakuvad veebirakenduse raamistikud, või kohalikus salvestusruumis, aga kuna tegemist on kahe rakendusega, siis esmaseks eesmärgiks on säilitada platvormidevahelist samaväärsust. Viimase tagab POST-päring serverile.

Restorani jaoks oli oluline see, et tellimuse staatus oleks reaalajas jälgitav ja muudetav. Selle funktsionaalsuse arendamiseks ei saa tarvitada tavalisi HTTP-protokolli päringuid eesrakenduse ja tagarakenduse vahel, mis kasutavad GET, POST, PATCH, DELETE meetodit. Üks võimalik lahendus on *WebSocket* tehnoloogia, mis kuulub Nest.js raamistikku. Nimetatud tehnoloogia põhineb sellel, et eesrakenduse ja tagarakenduse vahel luuakse pidev sidekanal, mis omakorda võimaldab koheselt teavitada restorani töötajat uuest tellimusest ja ilma lehte värskendamata näha tellimuse tegelikku staatust.

Selle tehnoloogia alternatiivideks on *Short Polling* või *Long Polling* tehnoloogiad, mille puhul tehakse päring iga määratud ajalise intervalli tagant. Nende variantide puuduseks on suur serveri koormamine.

3.2.1 Turvalisus

Kõik kasutaja päringud peaksid olema kaitstud kolmandate isikute sekkumise eest. Turvalisuse tagamiseks kasutatakse tagarakenduses JSON Web Token standardit. JSON Web Token (JWT) on avatud standard RFC-numbriga 7519, mis paneb aluse kompaktsele ja iseseisvale viisile osapoolte vahelise teabe turvaliseks edastamiseks JSON-objekti abil [9]. Edastatud teavet saab kontrollida ja usaldada, kuna see on digitaalselt allkirjastatud. JWT-sid saab allkirjastada salajase või avaliku/privaatse võtme paariga, kasutades krüpteerimise algoritme. Lisaohutuse saavutamiseks kodeeritakse registreerimise hetkel kasutaja salasõna räsiks, millele lisatakse nn. soola (inglise keeles *Salt*[1]), mida omakorda võetakse *.env* [2] failist.

Kõigil rakenduses olevatel API teedel on oma ligipääsu tasand, mis määratakse, kasutades "valvuri klassi" e. *Role guard*i. Rakenduses on neli kasutajarolli: anonüümne kasutaja, *user* e. registreeritud kasutaja, *staff* e. restorani töötaja ja *admin* e. administraator. Tänu rolli valvurile saab korraldada suhtlust tagarakendusega, kus toimub kasutaja tegevuste piiramine vastavalt nende ligipääsu tasandile, paindlikul ja turvalisel viisil. Üheks näidiseks on API teed, mis vastutavad toodete loomise, redigeerimise ja kustutamise eest. Iga sellise kontrolleri tee algoritme on "dekoraator"[3] @Roles('admin'), mis tähendab, et seda kontrollerit saab kasutada ainult administraatori rolliga kasutaja (Joonis 2).

[1] Salt – juhuslik andmehulk, mis lisatakse enne räsimist paroolile ning salvestatakse koos räsiga https://et.wikipedia.org/wiki/Sool_(kr%C3%BCptograafia)

[2] .env – tekstikonfiguratsioonifail rakenduste keskkonna konstantide juhtimiseks

[3] Dekoraator - funktsioon, mis võtab teise funktsiooni ja laiendab viimase funktsiooni käitumist ilma seda selgesõnaliselt muutmata https://medium.com/google-developers/exploring-es7-decorators-76ecb65fb841

```
@Roles('admin')
@Post()
async addMenuItem(
  @UploadedFile() image: Express.Multer.File,
  @Body() dto: CreateMenuItemDto,
): Promise<MenuItem> {
```

Joonis 2 Rolli valvuri kasutamise näidis

3.3 Andmebaasi struktuuri analüüs

Loodud andmebaasi struktuur koosneb 15 tabelist (Lisa 2).
User tabelis salvestatakse kasutaja andmed. Neid on kahte tüüpi:

1. andmed, mis eksisteerivad ühes eksemplaris ja hoitakse *User* tabeli sees: ees- ja perekonnanimi, telefon, e-post

2. andmed, mis eksisteerivad mitmuses ja on *User* tabeliga seotud võõrvõtmete paariga: kasutaja aadressid (tabel *Address*) ja maksmismeetodid (tabel *PaymentCard*) (andmebaasi tasandil üks-mitmele seos)

Kasutaja loomise protsessis seotakse kasutaja ühe ostukorviga (andmebaasis *Cart*), mida hiljem kasutatakse toodete soetamiseks toodete ostukorvi lisamise protsessis ja kasutaja ostukorvi kogusumma saamiseks.

Ostukorvis asuvad tooted hoitakse sidetabelis *MenuItemInCart*. Selles tabelis seotakse üks toode ühe kasutaja ostukorviga ning kuna kasutaja saab lisada mitu ühe tüübi toodet, siis selle jaoks on tabelis olemas veerg *quantity*.

Sarnaselt toodete lisamise tabelile on andmebaasis sidetabel *MenuItemInFavorites*, mis sisaldab kasutaja poolt lemmikute nimekirja pandud tooteid.

Kui kasutaja kustutab kas oma aadressi, maksmismeetodi, toote ostukorvist või lemmikustest, siis need read kustutatakse andmebaasist ära.

Andmebaasis on ka staatilised tabelid, mille sisu kas muutub väga harva (üheks põhjuseks saab äriloogika muutmine) või ei muuda üldse. Need tabelid on:

- *Role* – kasutaja roll süsteemis, väärtused on: *User*, *Staff* ja *Admin*
- *PaymentMethod* – võimalikud maksmismeetodid
- *DeliveryMethod* – võimalikud kohaletoimetamise viisid
- *OrderStatus* – võimalikud tellimuse staatuse olekud

Neid tabeleid ei täideta manuaalselt, vaid kasutatakse nn. *seed* [4] skripti rakenduse käivitamisel. Kõikidel sellistel tabelitel on teiste tabelitega üks-mitmele seos ja nende kasutamine on vajalik andmete filtreerimiseks ja haldamiseks. Nii saab näiteks restorani töötaja eraldada kõik tellimused staatusega 'ootel'.

Kõik kasutaja tellimused hoitakse tabelis *Order*. *Order* tabelis salvestatakse tellimuse kuupäev; kirjeldus, mis genereeritakse ostukorvis asuvate toodete põhjal; kasutaja kontaktandmed; tellimuse summa koos kohaletoimetamise tasuga ja võõrvõtmed ülaltoodud staatilistele andmetele. Tellimuse loomise protsessis ostukorvis asuvatest toodetest tehakse koopia tabelisse *MenuItemInOrder*, mis on vajalik selleks, et kasutaja saaks soovi korral tellimust korrata samade toodete ja kogustega. Tellimuse lõpulejõudmisel on kasutajal võimalus anda restoranile tagasidet, mille jaoks on andmebaasis tabel *Feedback*. Kuna kasutaja võib unustada või mitte soovida tagasisidet anda, siis on tagasiside veerg tabelis tühistav (inglise keeles *nullable*).

Kõik tabelid saab jagada kolme kategooriasse:

1. Aktiivsed – andmete salvestamine, muutmine ja kustutamine toimub tihti
2. Keskmise aktiivsusega - andmete salvestamine, muutmine ja kustutamine toimub aeg-ajalt
3. Madala aktiivsusega – andmete salvestamine, muutmine ja kustutamine toimub harva

Aktiivsete tabelite esindaja on tabel *MenuItemInCart*, kuna toodete lisamine ja kustutamine kasutaja poolt toimub väga tihti. Keskmise aktiivsusega on tabelid *MenuItemInFavorites*, *MenuItemInOrder* ja *Order*. Madala aktiivsusega on kõik teised.

[4] *Seeding* – protsess, mille ajal täidatakse andmebaasi esialgse andmekogumisega

3.4 Ülevaade kasutatud tehnoloogiatest

3.4.1 Andmebaas

Andmebaasiks oli valitud tasuta ja avatud lähtekoodiga relatsioonandmebaas PostgreSQL, kuna rakenduses loodavad andmed on omavahel ühendatud ja neil on fikseeritud andmemall. Relatsioonandmebaasi suureks plussiks on andmetüüpide rangus, mis vähendab vigade esinemist tulevikus. Võrreldes teiste relatsioonandmebaasidega, Postgres rakendab mitmeversioonilist samaaegsuse juhtimist (MVCC) ilma lugemislukkudeta, toetab paralleelseid päringuplaane, mis võivad kasutada mitut protsessorit/tuuma, saab luua indekseid mitteblokeerival viisil. Postgres on tuntud andmete terviklikkuse kaitsmise poolest tehingu tasemel. See muudab selle andmete kahjustamise suhtes vähem haavatavaks [11].

Maksmissüsteemi toetamiseks kasutati Stripe'i. Stripe on ettevõte, mis pakub peamiselt maksete töötlemise tarkvara ja rakendusliideseid (API-sid) e-kaubanduse veebisaitidele ja mobiilirakendustele. Stripe'i suureks eeliseks on see, et see annab võimaluse siduda loodava rakenduse andmebaasi Stripe'i andmebaasiga ja delegeerida klientide maksmiskaartide validatsiooni ning maksmise protsessi ise.

3.4.2 Tagarakendus

Nest.js on üks kiiremini kasvavaid Node.js-i raamistikke tõhusate ja ettevõtte tasemel tagarakenduste loomiseks. Nest.js on loodud kaasaegse *Typescripti* baasil, mis võimaldab arendajatel luua skaleeritavaid, kergesti hooldatavaid ja testitavaid rakendusi, järgides SOLID ja 12-faktoriliste rakenduste põhimõtteid [12]–[13]. Andmebaasiga töötamiseks integreeriti Sushihubi tagarakendusse Prisma ORM, mis võimaldab genereerida tüübikindlaid olemite mudeleid andmebaasi mudeli põhjal, teha migratsioone ja kasutada tüübikindla päringu koostajat. Lisaks sellele on Prisma sees sisseehitatud kasutajaliides, mis on tuntud PhpMyAdmin'i alternatiiv.

3.4.3 Mobiilirakendus

Mobiilirakendust saab tänapäeval arendada kahel viisil: kas platvormipõhilise keelega nagu Kotlin/Java (Androidi jaoks) ja Swift (IOS jaoks) või kasutada raamistikke, mis võimaldavad luua nn platvormideüleseid rakendusi. Üheks populaarsemaks platvormideüleseks raamistikuks on Facebooki poolt loodud *React Native*. *React Native*

on *JavaScripti* raamistik natiivselt renderdavate mobiilirakenduste kirjutamiseks iOS-i ja Androidi jaoks. See põhineb *React*il, kasutajaliideste loomiseks kasutatud Facebooki *JavaScript*i teegil, kuid brauserite asemel on see suunatud mobiiliplatvormidele [14]. Arendamisprotsessi kiirendamiseks valis autor *Expo* raamistiku, mis on avatud lähtekoodiga raamistik rakendustele, mis töötavad Androidis, iOS-is ja veebis. *Expo* ühendab parimad võimalused mobiili ja veebi jaoks ning toetab paljusid olulisi funktsioone rakenduse koostamiseks ja skaleerimiseks, nagu reaalajas värskendused, rakenduse kohene jagamine ja veebitugi [15].

3.5 Ülevaade olemasolevatest rakendustest

Enne käesoleva lõputöö praktiliseks osaks oleva mobiilirakenduse arendamist analüüsiti ka olemasolevate toidutellimise rakenduste funktsionaalsust. Valiti kolme Eestis toimiva ettevõtete mobiilirakendused: MacDonalds, DodoPizza ja SushiKing. Neist viimane võeti otsese võrdluse tegemiseks, kuna ettevõte tegutseb samas tootevaldkonnas.

McDonaldsi rakenduse kasutamiseks on vaja registreerida, kasutades oma e-posti. Rakenduse kodulehel on interaktiivsed infotahvlid uudiste ja sooduspakkumistega. Eesmärgipärase interaktsiooni tagamiseks on rakendusel navigeerimismenüü, mis koosneb järgmistest valikutest:

- “Deals” - praegused pakkumised
- “Order” – tellimuste ajalugu
- “Food” – iga menüüs asuva toidu kirjeldus, toiteväärtuse informatsioon ja allergeenide nimekiri
- “Restaurants” – kaart lähimate restoranidega

„Deals” ekraanile minnes avaneb praeguste pakkumiste nimekiri, mille hulgas on eripakkumised ja sellised tooted, mille puhul on tellimine võimalik ka lojaalsusprogrammis “Points” kogutud punktide abil.
Punkte koguneb iga ostu eest ja punktide hulk sõltub tellimuse summast. Toidu kohaletoimetamise võimalused puuduvad, tellimuse kättesaamine on võimalik ainult restoranist.

DodoPizza rakenduse kasutamiseks on vaja registreerida, kasutades oma telefoninumbrit. Eesmärgilise interaktsiooni tagamiseks on rakendusel olemas navigeerimismenüü, mis koosneb järgnevatest valikutest:

- “Menüü” – rakenduse koduleht, mis sisaldab interaktiivseid tahvleid eripakkumistega ja ettevõtte uudistega. Selle alaosas asub sektsioon „Uus ja populaarne“, kus reklaamitakse uusi ja kasutajate hulgas populaarseid tooteid. Omakorda selle sektsiooni all on toodud kogu nimekiri pakutavatest toitudest koos kategooriate valikuga
- “Profiil“ – kasutaja profiil, kus on võimalik vaadata tellimuste ajalugu, määrata kohaletoimetamise aadresse, sealsamas kuvatakse ka “Dodo-raha“ kogust.

DodoPizza lojaalsuse programm on loodud niimoodi, et iga kulutatud viie euro eest laekub kasutaja kontole üks „Dodo-coin", mille saab vahetada sõltuvalt müntide kogusest teatud toitude vastu

- "Kontaktid" – ekraan, millel saab vaadata restoranide asukohti ja võtta restoraniga ühendust
- "Ostukorv" – lisatud toitude nimekiri. Saab valida toidu kohaletoimetamise või minna ise restorani järgi

SushiKingi rakendust saab kasutada ka registreerimata. Tegemist on põhimõtteliselt ettevõtte veebikodulehe mobiiliversiooniga, mis on tehtud eraldi rakenduseks. Rakenduse avamisel ilmub koduleht. Kodulehe ülaosas asub sektsioon uudisteste ja pakkumistega. Allpool kuvatakse enimmüüdud toitude nimekiri, siis restorani menüü. Menüü koosneb kategooriate linkidest, mis viivad vastava kategooria toitude nimekirjale. Menüüst allpool asub tarneinfo.
Rakendusel on lojaalsusprogramm, mis põhineb nn. *cashback*[5]il, andes võimaluse saada järgmiselt ostult 15% raha tagasi. Lisaks on rakenduses ka sooduskupongid.
Rakenduse igal ekraanil kuvatakse päis, mis koosneb ostukorvist ja menüünupust. Vajutades päises olevat menüünuppu, ilmub ekraan, kus kasutaja saab vaadata oma konto infot, tellimuste ajalugu, lojaalsusprogrammi kontojääki, sooduskuponge, valida rakenduse keelt ja enda asukohalinna. Sellest allpool asub toitude kategooriate nimekiri.

Järgnevas tabelis on võrreldud rakenduse Sushihub funktsioone, mis olid tellija poolt nõutud, ülaltoodud olemasolevate rakendustega. Pluss-märgiga tähistatakse kirjeldatud funktsiooni olemasolu ja miinus-märgiga puuduvaid.

Funktsioon	Sushihub	McDonalds	DodoPizza	SushiKing
Registreerimine ja autoriseerimine	+	+	+	+
Toodete loendi vaatamine	+	+	+	+
Filtreerimine toidu kategooria järgi	+	-	+	+
Toidu koostise info vaatamine	+	+	+	+
Toidu lisamine lemmikutesse	+	-	-	-
Toidu lisamine ostukorvi	+	+	+	+

[5] Cashback - preemiaprogramm, kus kliendid saavad ostlemisel kulutatud raha tagasi teenida

Funktsioon	Sushihub	McDonalds	DodoPizza	SushiKing
Tellimuse esitamine	+	+	+	+
Tellimuste ajalugu vaatamine	+	+	+	+
Tellimuse reaalaja staatuse jälgimine	+	+-	+-	+-
Kehtivate pakkumiste vaatamine	+	+	+	+
Kaardimakse	+	+	+	+

Tabel 1. Arendatud mobiilirakenduse funktsionaalsuse võrdlus olemasolevate rakendustega

Ülaltoodu järgi saab teha järelduse, et mõned funktsioonid võrreldavates rakendustes kas puuduvad või on implementeeritud osaliselt:

1. lemmikutesse lisamise võimalus puudub võrreldavates rakenduses täielikult
2. tellimuse staatuse reaalajas jälgimine on võrreldavates rakenduses implementeeritud osaliselt - selleks näidatakse kasutajale hinnangulist tellimuse teostusaega, kuid ei näidata, millises etapis on tellimus

4 Arendustegevuse kirjeldus

4.1 Tagarakenduse arendamine

Tagarakenduse arendamine viidi läbi ühiselt teise lõpetaja, Martin Aleksander Kostjajeviga. Autori ülesanne oli kasutajate autentimis/autorisatsiooni loogika arendamine, rakenduse turvalisuse tagamine ja tellimuse staatuse uuendamise funktsiooni arendamine. Selleks sai kasutusele võetud Nest.js raamistik, mis võimaldas arendada REST API, millega tagatakse veebirakenduse ja mobiilirakenduse vaheline suhtlus HTTP protokolli kaudu.

4.1.1 Tagarakenduse projekti struktuur

Tagarakenduse projekti struktuur on jagatud erinevatesse kaustadesse. Iga kaust vastutab ühe rakenduse osa eest (Lisa 6):

- „*auth*" sisaldab komponente ja teenuseid, mis vastutavad rakenduse autentimise eest
- „*constants*" sisaldab kõiki muutumatuid väärtusi ja nimetusi, mida kasutatakse esmaste andmete genereerimiseks ja edasiseks töötlemiseks teenustes
- „*controllers*" sisaldab kõiki kontrollereid
- „*core*" sisaldab kõiki abstraktsiooniklasse
- „*dal*" sisaldab andmetele kättesaadavuse kihti
- „events" sisaldab *WebSocket*it
- „*expection-filter*" sisaldab tagarakenduses kasutatavat vahevara (inglise keeles *middleware* [6]) erandite püüdmiseks
- „*models*" sisaldab andmebaasi olemite mudeleid
- „*modules*" sisaldab klasse, mida Nest.js kasutab rakenduse struktuuri korraldamiseks
- „*routes*" sisaldab API marsruuteri konfiguratsiooni
- „*services*" sisaldab olemasolevaid teenuseid, mis loovad teenuste kihi
- „*utils*" sisaldab abifunktsioone, mida kasutatakse teenustes

[6] Middleware - funktsioon, millel on kogu juurdepääs objekti taotlemiseks, objektile vastamiseks ja rakenduse päringu-vastuse tsüklis https://en.wikipedia.org/wiki/Multiversion_concurrency_control

4.2 Mobiilirakenduse arendamine

Mobiilirakenduse arendamine algas prototüübi loomisega *Figma* [7] keskkonnas koostöös kliendiga. Kui prototüüp heaks kiideti, alustati selle põhjal mobiilirakenduse arendamist.

4.2.1 Mobiilirakenduse projekti struktuur

Mobiilirakenduse projekti struktuur on jagatud erinevatesse kaustadesse. Iga kaust vastutab ühe rakenduse osa eest (Lisa 7):

- „*assets*" sisaldab kasutatavaid ikoone
- „*components*" sisaldab mobiilirakenduse ekraanides kasutatavaid komponente;
- „*constants*" sisaldab muutumatuid väärtusi ja nimetusi
- „*context*" sisaldab mobiilirakenduse oleku juhtimise komponente
- „*hooks*" sisaldab korduvkasutatavad komponente, mis ilmuvad erinevates ekraanides
- „models" sisaldab andmebaasis olevaid mudeleid
- „*navigation*" sisaldab marsruutide konfigureerimiskomponente
- „*screens*" sisaldab mobiilirakenduse ekraane
- „*services*" sisaldab tagarakendusega suhtlemise teenuseid
- „*types*" sisaldab abimooduleid, mis võimeldavad TypeScript kompilaatoril töötada *.env*- ja pildifailidiega
- „*utils*" sisaldab abifunktsioone, mida kasutatakse komponentides

[7] Figma - koostööl põhinev veebirakendus liidese kujundamiseks https://en.wikipedia.org/wiki/Figma_(software)

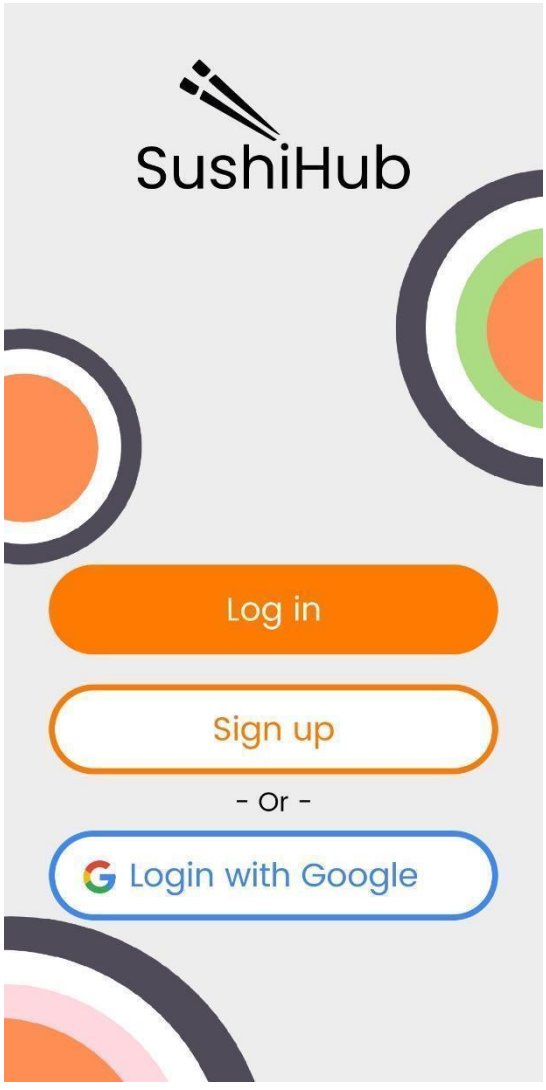
5 Tulemused

Käesolevas peatükis kirjeldatakse valminud mobiilirakendust, selle funktsionaalsust. Lisaks tuuakse välja võimalikud tulevikus tehtavad täiendused.

5.1 Ülevaade rakendusest

5.1.1 Avaekraani vaade

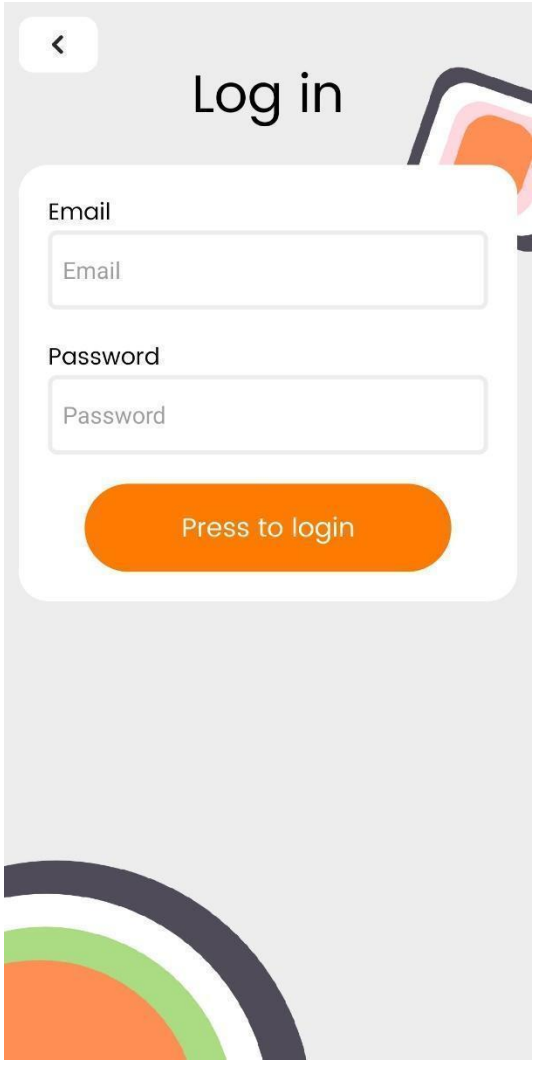
Mobiilirakenduse avamisel kuvatakse kasutajale tervitusekraan, kus on restorani logo ja kolm nuppu, mis vastutavad kasutaja autentimise eest.



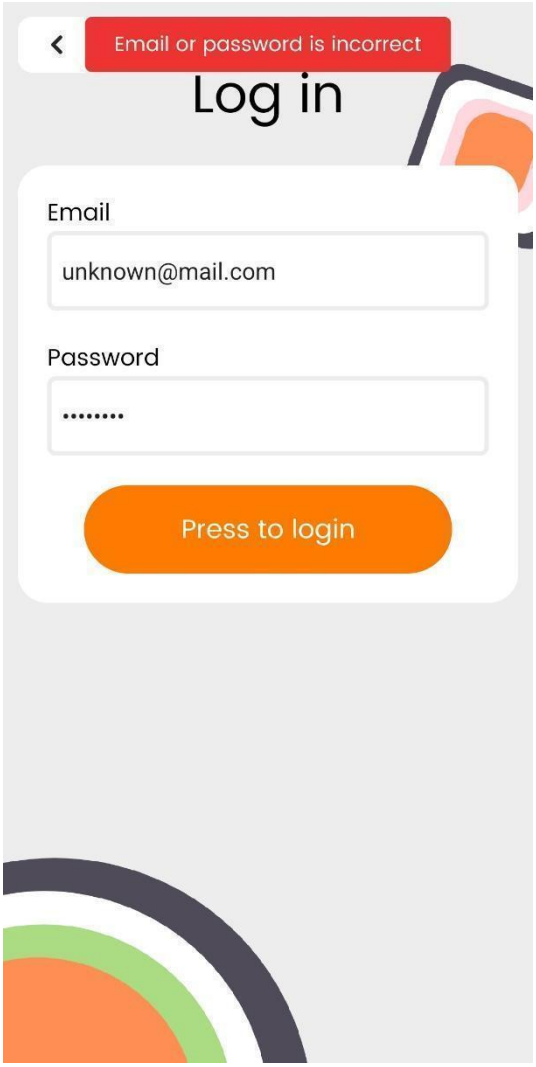
Joonis 2. Tervitusekraan

5.1.2 Sisselogimisvaade

Kui kasutaja vajutab nupule “Log in”, siis suunatakse ta sisselogimisekraanile. Rakenduses autentimiseks peab kasutaja sisestama oma e-postiaadressi ja loodud salasõna. Kui sisestatud isikutunnustega kasutajat ei eksisteeri, siis ilmub ekraani ülaosas vastava sisuga veateade. Selleks, et minna tagasi avaekraanile, on üleval vasakus nurgas nupp noolega. Selline tagasinupu asukoht on hästi tüüpiline mobiilirakenduste jaoks, sest paljud mobiilirakenduste disaini põhimõtted üldiselt laenatakse veebibrauseri disainist [10].



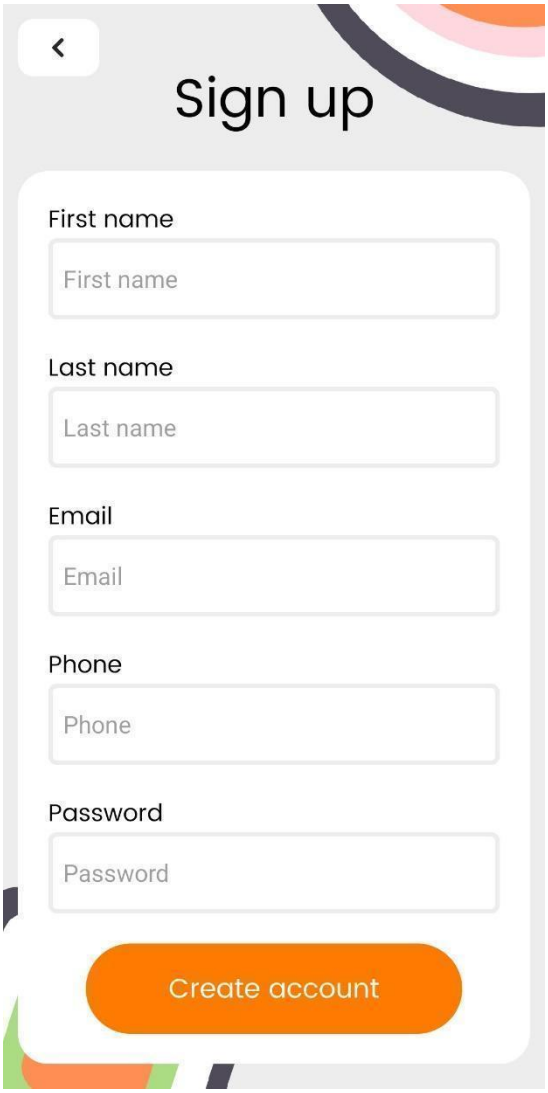
Joonis 3. Sisselogimise ekraan



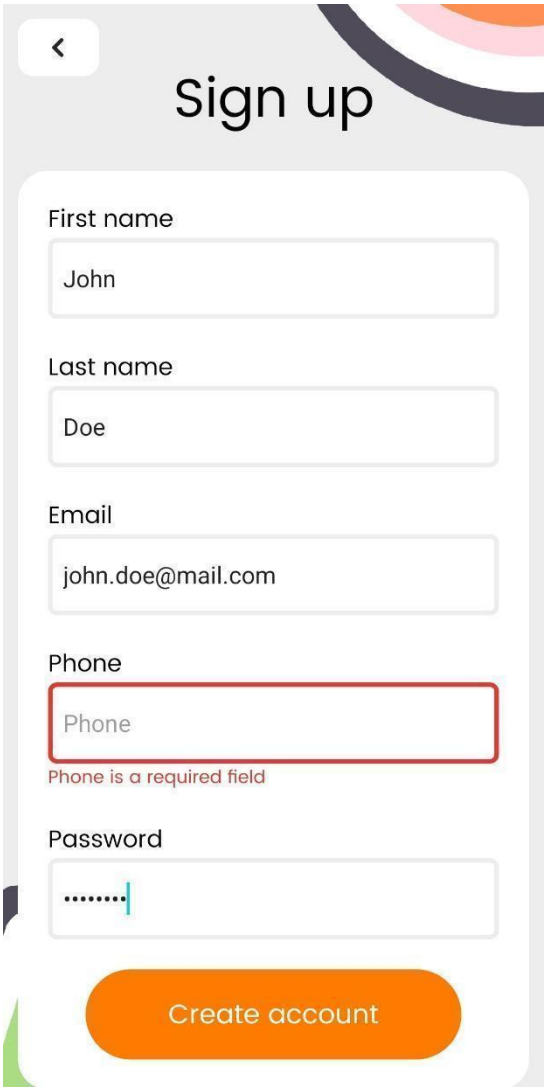
Joonis 4. Vea teave näidis, kui registreerimata kasutaja proovib logida sisse

5.1.3 Registreerimisvaade

Kui kasutaja vajutab nupu “Sign up” peale, siis ta suunatakse registreerimisekraanile. Rakenduses registreerimiseks küsitakse kasutajalt tema ees- ja perekonnanime, e-posti, telefoninumbrit ja salasõna. Võrreldes sisselogimisekraaniga kasutatakse selles vaates rakenduse kliendipoolset (*client-side*) valideerimist, mis ei luba edastada registreerimisvormi enne, kui kõik väljad on täidetud.



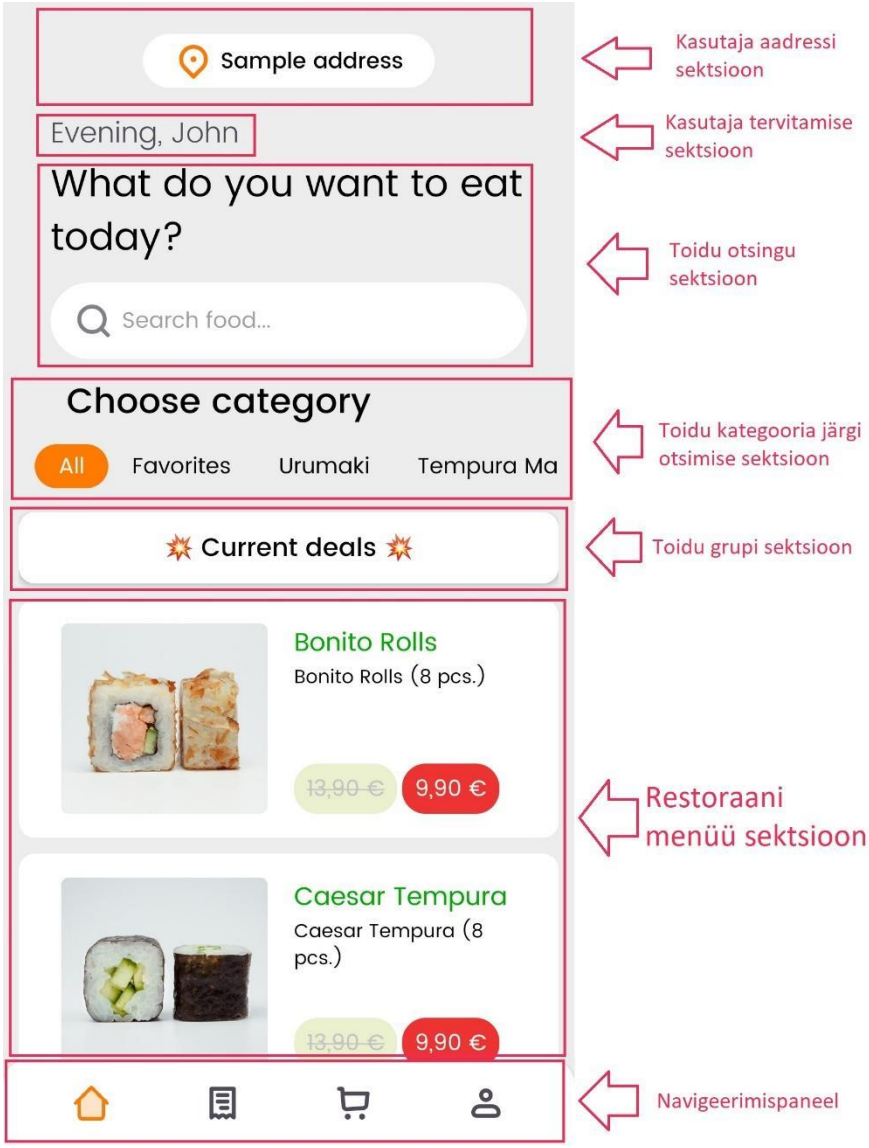
Joonis 5. Registreerumisekraan



Joonis 6. Veateade tühjaks jäetud välja puhul

5.1.4 Koduleht

Pärast edukat sisselogimist või registreerimist kuvatakse rakenduse koduleht. Koduleht koosneb mitmest sektsioonist: kasutaja vaikimisi aadress, kasutaja tervitamise sõnum, toidu otsing roa nime järgi, toidu otsing roa kategooria järgi, menüü, navigeerimispaneel.



Joonis 7. Rakenduse koduleht

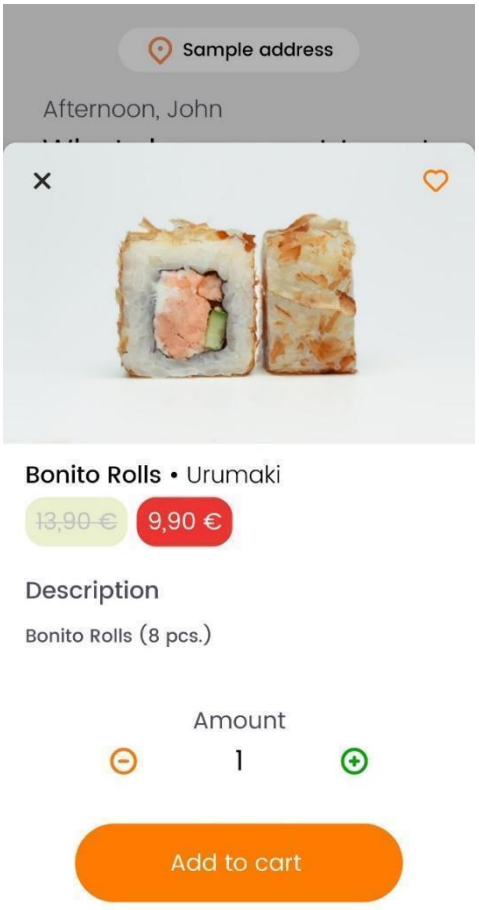
5.1.5 Navigeerimispaneel

Navigeerimispaneel on ainuke komponent rakenduses, mis on nähtav igal ekraanil. Navigeerimispaneel koosneb neljast nupust, mis viivad kasutaja erinevatele ekraanidele: koduleht, tellimuste ajalugu, ostukorv ja kasutaja profiil. Aktiivse ekraani näitamiseks navigeerimispaneelil tõstetakse vastav nupp värviliselt esile. Navigeerimispaneeli positsioon ekraani alaservas teeb ekraanide vahel liikumise lihtsamaks, kuna asub inimese pöidla ulatuses [10].

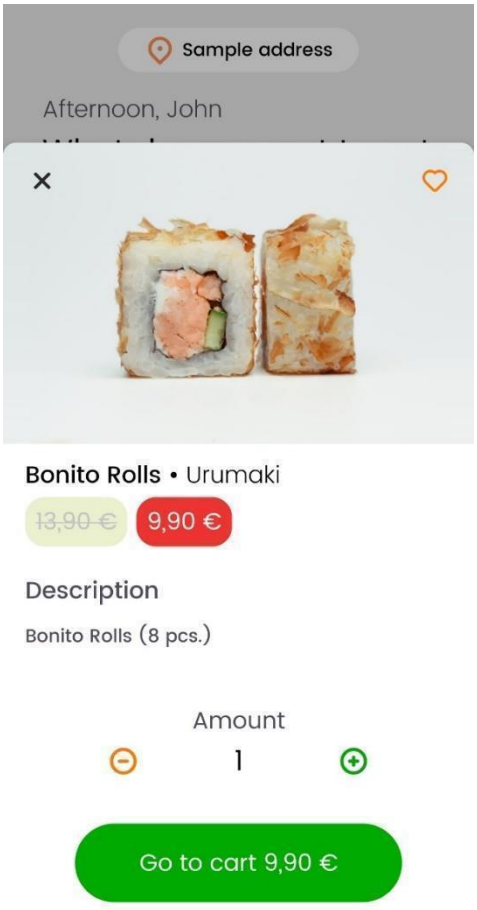
Joonis 8. Navigeerimispaneel

5.1.6 Toote detailide vaade

Kui kasutaja klõpsab roa peale, siis ekraani alaservast hüppab üles roa detailne vaade, kus saab roa kohta täpsemat infot, südamekujulise nupu abil lisada selle lemmikutesse ning soovitud koguse sisestamise järel lisada roa ostukorvi (Joonis 8). Roa korvi lisamisel muutub nupp “Add to cart” roheliseks ja selle sees kuvatakse ostukorvi summa (Joonis 9). Kui kasutaja vajutab selle nupu peale. suunatakse ta ostukorvi ekraanile.



Joonis 9. Roa detailide vaade



Joonis 10. Roa detailide vaade, kui roog lisati ostukorvi

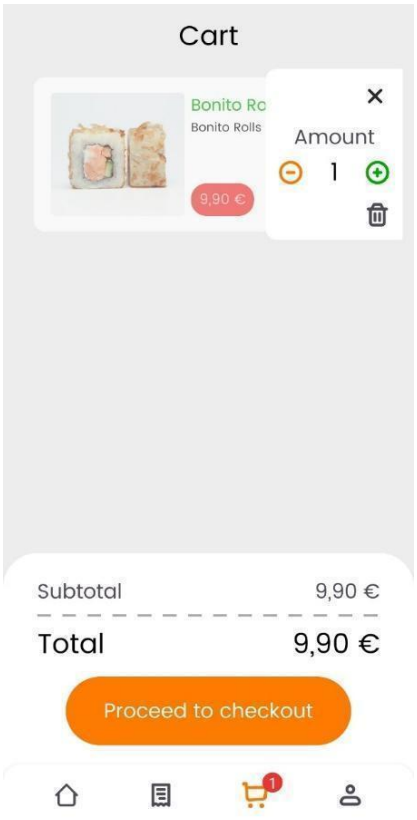
Kodulehel tähistatakse juba ostukorvi lisatud road rohelise vertikaalse joonega tootekasti vasakus servas (Joonis 10).

Joonis 11. Ostukorvi lisatud roa tähistus

5.1.7 Ostukorvi vaade

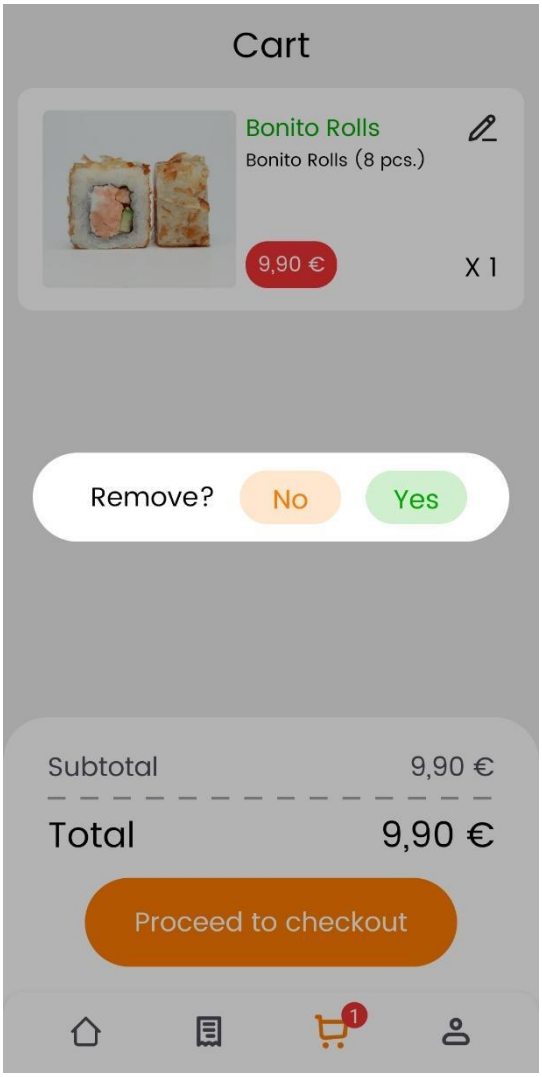
Kasutaja saab navigeerida ostukorvis kahel viisil: kas minna roa detailivaatesse või kasutada navigeerimispaneeli. Ostukorvis kuvatakse kõik lisatud road nende kogustega ja allpool on esitatud ostukorvi summa. Roa kasti paremas servas on pliiatsi ikooniga nupp, millele vajutades ilmub paneel, kus saab muuta roa kogust või kustutada selle ostukorvist.

Joonis 12. Ostukorvi vaade



Joonis 13. Roa koguse muutmise paneel

Kui kasutaja muudab roa kogust, on võimalik, et ta vajutab kogemata prügikasti ikooni peale. Roa tahtmatult eemaldamise vältimiseks järgneb kustutusnupu vajutusele kinnitusaken. Pärast toote kustutamist ostukorvist ilmub ekraani ülaserva vastav teade.



Joonis 14. Kinnitusaken roa kustumisel

Joonis 15. Roa kustutamise teavitus

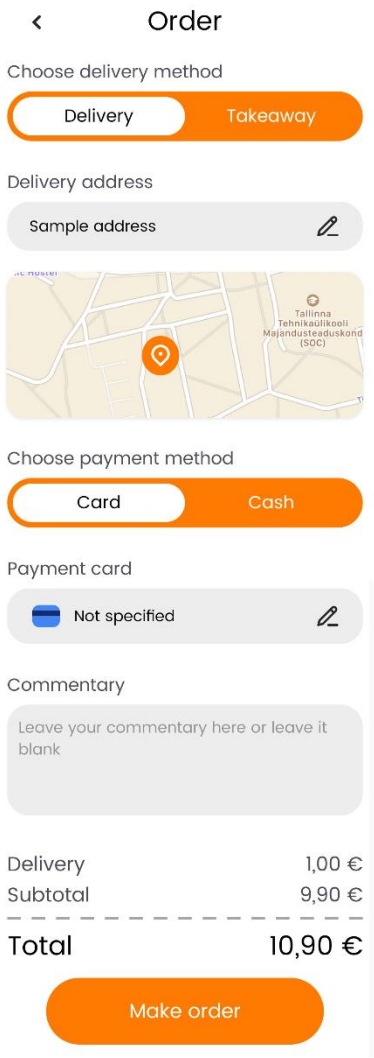
5.1.8 Ostukorvi aktiivsuse kuvamine navigeerimispaneelil

Roa lisamisel ilmub navigeerimispaneelil asuva ostukorvi ikooni kohale punane ring lisatud roogade kogusega. See on populaarne viis juhtida kasutaja tähelepanu tema poolt sooritatud tegevustele rakenduses [10].

Joonis 16. Ostukorvi ikoon lisatud roogade puhul

5.1.9 Tellimuse vormistamise vaade

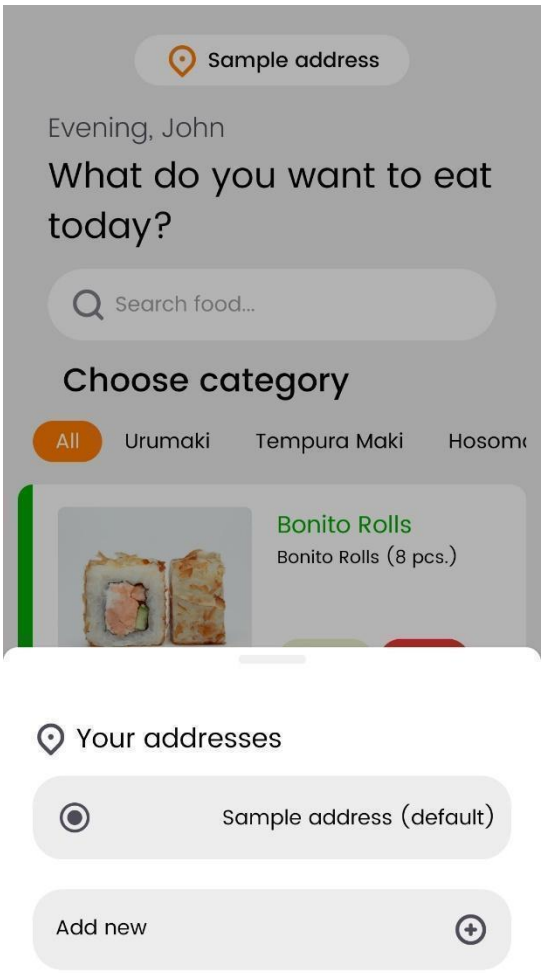
Kui kasutaja on valmis oma tellimust esitama, siis vajutab ta ostukorvis asuva nupu “Proceed to checkout” peale ning ta suunatakse tellimuse vormistamise ekraanile. Siin on kasutajal võimalik valida talle sobiv tarnemeetod, maksmisviis ja soovi korral lisada tellimusele kommentaar (Joonis 16). Samamoodi nagu registreerimisekraanil, ei saa liikuda järgmisele lehele ehk antud juhul tellimust esitada enne, kui kõik vajalikud väljad on täidetud.



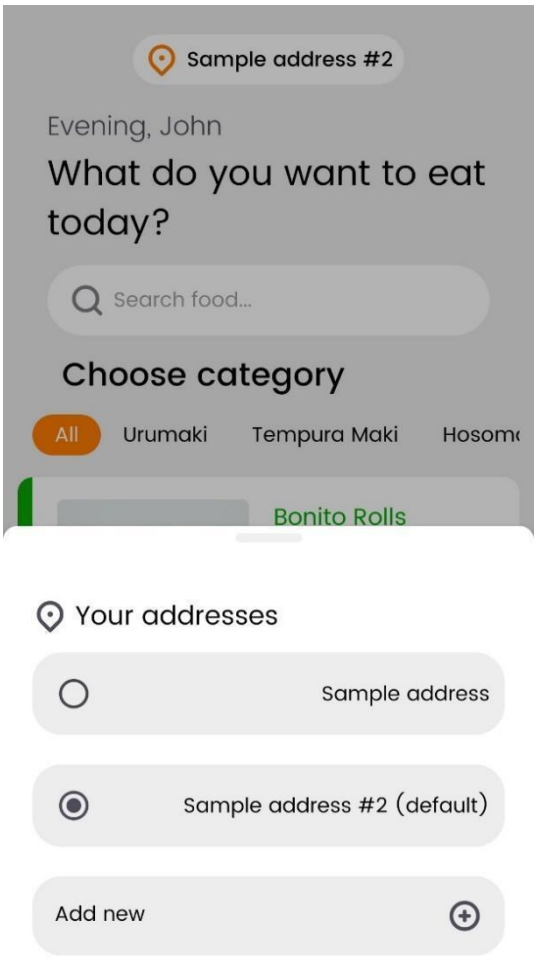
Joonis 17. Tellimuse vormistamise vaade

5.1.10 Aadressi ja maksmismeetodi vaade

Rakenduse kasutajal on võimalik siduda oma kontoga mitu tarneaadressi ja maksmismeetodit. Nende sätete muutmine toimub ekraani alaservast ilmuvas aknas, tuntud ka kui alumine leht (inglise keeles *bottom sheet*) (Joonis 17). Kuna lisada saab mitu aadressi ja makseviisi, siis tuleb kindlaks teha, millist neist soovib kasutaja praegusel hetkel oma tellimuse jaoks kasutada. Selle tarbeks on iga aadressi kõrval raadionupp, mille peale vajutades muutub valitud aadress vaikimisi aadressiks (Joonis 18).



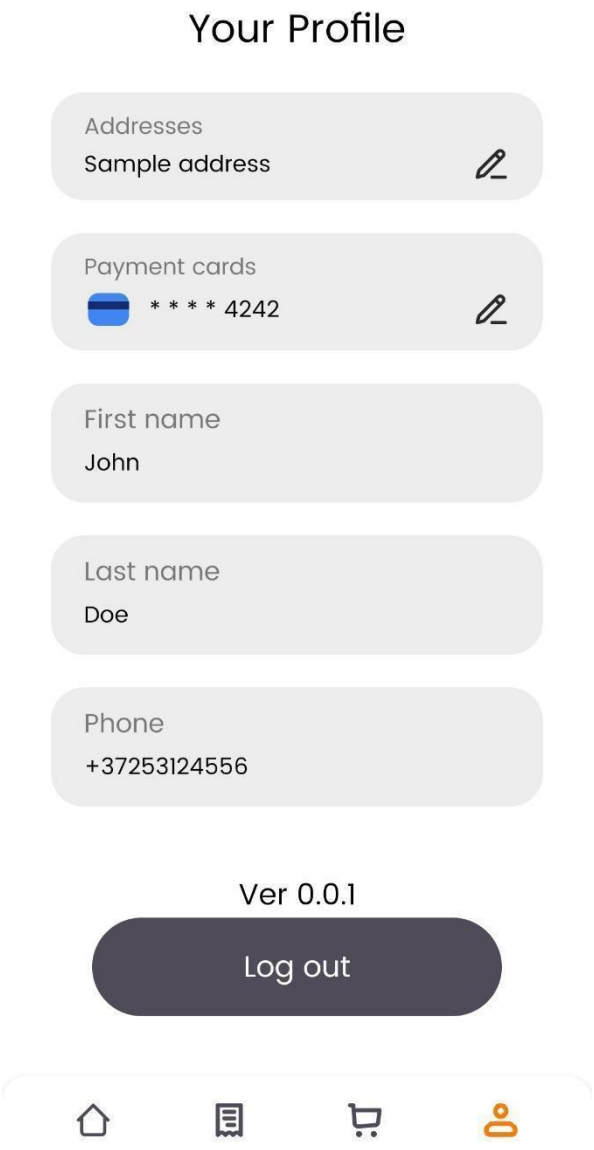
Joonis 18. Aadressi lisamise vaade



Joonis 19. Raadionupule vajutades vaikimisi aadressi muutmine

5.1.11 Profiili vaade

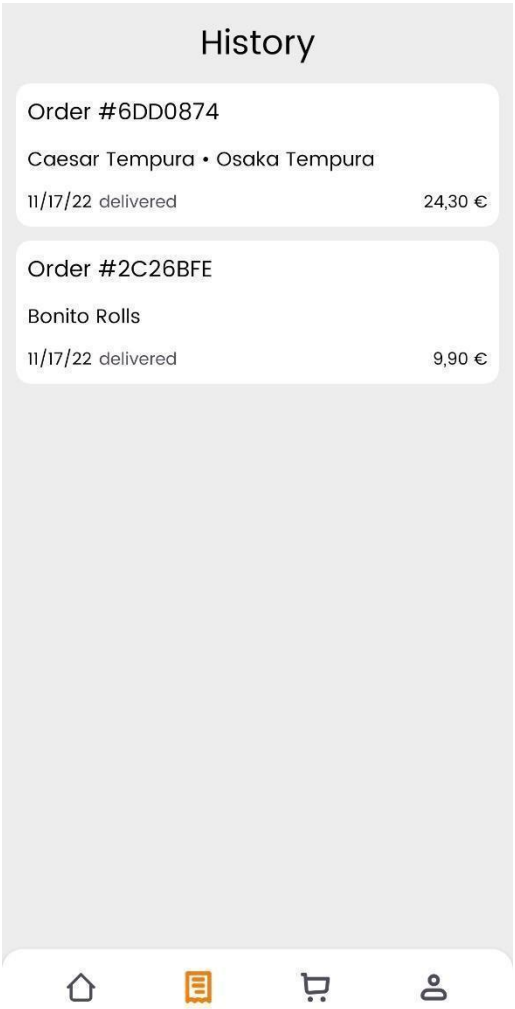
Profiilivaates saab kasutaja muuta nime, mida kuvatakse tema tellimustel ja telefoninumbrit, lisada uusi või kustutada vanu aadresse ja maksmismeetodeid. Lisaks sellele on ekraani alaservas väljalogimisnupp ja kuvatud rakenduse versioon.



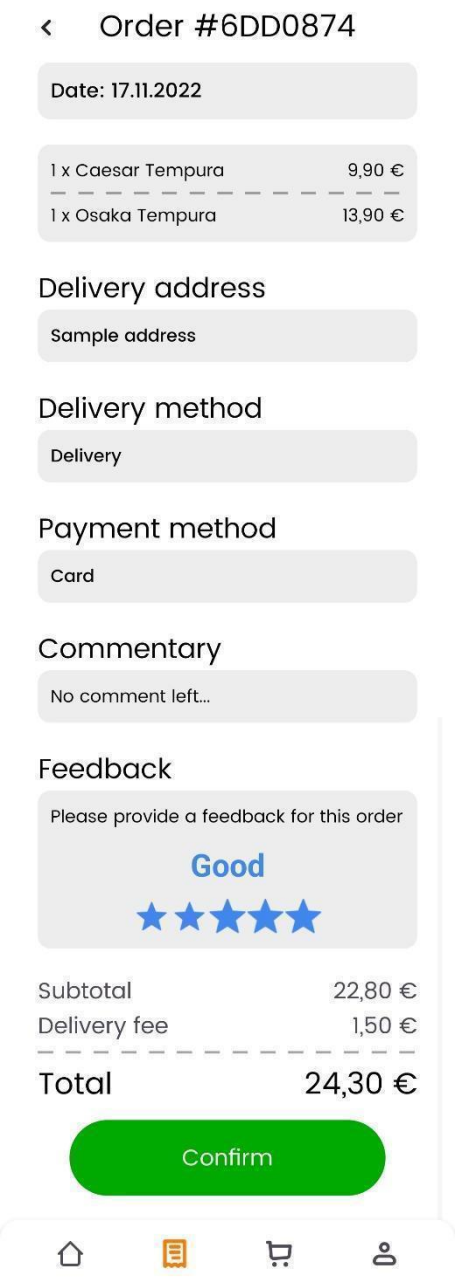
Joonis 20. Profiilivaade

5.1.12 Tellimuste ajaloo vaade

Tehtud tellimuste vaates näeb kasutaja loendit kõigist oma sooritatud tellimustest ja soovi korral vaadata iga tellimuse üksikasju, vajutades loendi ühe kande peale. Üksikasjalikus vaates saab tellimust viiepallisel skaalal hinnata ja saata seeläbi restoranile tagasisidet.



Joonis 21. Tellimuste ajaloo vaade



Joonis 22. Tellimuse üksikasjade vaade

5.2 Võimalikud edasiarendused

Mobiilirakenduse arendamine oli jagatud etappideks ja esimeses iteratsioonis oli vaja implementeerida ainult tellija poolt esitatud funktsionaalsus. Mobiilirakendus sai valmis ja järgmiseks etapiks on testimise faas, millele järgneb mobiilirakenduse postitamine Google Play ja Apple Store platvormidele.

Olemasolevateste rakenduste ülevaatest oli valitud funktsionaalsus, mida hakatakse implementeerima järgmistes iteratsioonides:

- kohalik valuuta ja sellega seotud sooduspakkumised
- ettevõtte uudiste sektsioon
- keelevalik
- ettevõtte kontaktide ekraan

Rakenduse funktsioonidest saab eriti õnnestununa esile tõsta tellimuse staatuse reaalajas uuendamise, mis hetkel töötab paremini kui teistest rakendustes. Teisalt saab seda veel täiendada kaardi vaatega, mis kuvaks kulleri asukoha.

Rakenduses puudub hetkel kaheastmeline autentimine. Kinnitussõnumi saatmise süsteem SMS või e-posti teel suurendaks rakenduse turvalisust.

6 Kokkuvõte

Käesoleva lõputöö raames arendati sushirestoranile hübriid-mobiilirakendus ja tagarakendus, mis võimaldab klientidel tellida toitu koju ning restorani omanikel hallata toitude menüüd ja sooduspakkumisi ning tellimuse staatust.

Projekti arendamine käis koostöös tellijaga ja teise lõpetajaga. Valmis projekt koosneb kolmest osast: veebirakendus, mobiilirakendus ja ühine tagarakendus. Projekti ehitamisel kasutati järgmisi tehnoloogiaid:

- Nest.js – millel põhineb tagarakenduse osa
- PostgreSQL – mida kasutati andmebaasi mootoriks
- React Native – raamistik platvormideüleste mobiilirakenduste loomiseks

Tagarakenduse arendamise vastutusalad oli jagatud osadeks, autori panus oli suunatud mobiilirakenduse arendamisele, tagarakenduse turvalisuse ja tellimuse jälgmise loogika tagamisele, teine lõpetaja tegeles peamiselt CRUD-toimingute ja veebirakendusega.

Valmis sai rakendus, milles oli täidetud tellija nõued: kasutaja saab luua endale konto, tal on võimalus vaadata toodete loendit koos sooduspakkumistega ja filtreerida tooteid kategooria järgi, kasutajal on võimalus lisada toitu ostukorvi ja esitada tellimus ning tasuda talle sobiva makseviisiga, samuti saab ta valida tarneviisi.

Võrdluseks valitud konkurentide rakendustega kõrvutades on loodud rakendus parem kasutajakogemuse poolest, sest see võimaldab lisada tooteid lemmikutesse. Nii muutub ostlemistegevus üldiselt lühemaks ja mugavamaks. Lisaks saab rakenduses jälgida tellimuse olekut reaalajas - see funktsioon on teistes rakendustes implementeeritud osaliselt.

Lõputöö analüüsi osas kirjeldatakse tagarakenduse arhitektuuri ja selle loogikat, andmebaasi arhitektuuri, valikut teistest olemasolevatest rakendustest ja võrreldakse loodud rakendust selle valimiga.

Lõputöö viimases peatükis kirjeldatakse rakenduse võimalikke puudusi ja täiustatavaid osi. Pärast testimise faasi plaanitakse rakendus üles laadida levitamisplatvormidel nagu Google Play ja App Store.

Kasutatud kirjandus

[1] Tandon et al: Anushree Tandon, Puneet Kaur, Yogesh Bhatt, Matti Mäntymäki, Amandeep Dhir, „Why do people purchase from food delivery apps? A consumer value perspective,“ Journal of Retailing and Consumer Services, Volume 63, 102667, ISSN 0969-6989, 2021 [Võrgumaterjal]. Available: https://doi.org/10.1016/j.jretconser.2021.102667 . [Kasutatud 20 november 2022].

[2] Traynor et al: Mark Traynor, Shaniel Bernard, Andrew Moreo, Sorcha O'Neill, „Investigating the emergence of third-party online food delivery in the U.S. restaurant industry: A grounded theory approach,“ International Journal of Hospitality Management, Volume 107, 103299, 2022 [Võrgumaterjal]. Available: https://doi.org/10.1016/j.ijhm.2022.103299 . [Kasutatud 20 november 2022].

[3] Belkin, Märt: Märt Belkin. „Nüüd on Eestis uus toidutellimise võimalus. On see Boltist ja Woltist parem?,“ Raha Geenius, Apr. 2020 [Võrgumaterjal]. Available: https://raha.geenius.ee/rubriik/uudis/nuud-on-eestis-uus-toidutellimise-voimalus-on-see-boltist-ja-woltist-parem/ . [Kasutatud 20 november 2022].

[4] Rebane-Mäe, Liisa: Liisa Rebane-Mäe. „Eesti vanim pitsakett tõi turule uudse toidutellimise platvormi,“ Ärileht, Sep.2020 [Võrgumaterjal]. Available: https://arileht.delfi.ee/artikkel/91205253/eesti-vanim-pitsakett-toi-turule-uudse-toidutellimise-platvormi . [Kasutatud 20 november 2022].

[5] Kostjaev: Martin Kostjaev, „Veebirakenduse arendus restoranist toidu tellimiseks,“ Martin Kostjajev, 2022 [Võrgumaterjal]. Available: https://digikogu.taltech.ee/et/Item/aaeac46f-0bf8-4ed7-831e-89960811dcf8 . [Kasutatud 20 november 2022].

[6] Chan et al: T.K.H. Chan, X. Zheng, C.M.K. Cheung, M.K.O. Lee, Z.W.Y. Lee, "Antecedents and consequences of customer engagement in online brand communities", J. Mark. Anal, 2 (2) 2014, pp. 81-97 [Võrgumaterjal]. Available: https://doi.org/10.1057/jma.2014.9 . [Kasutatud 20 november 2022].

[7] Ray et al: Arghya Ray, Amandeep Dhir, Pradip Kumar Bala, Puneet Kaur, „Why do people use food delivery apps (FDA)? A uses and gratification theory perspective,“ Journal of Retailing and Consumer Services, Volume 51, pp. 221-230, ISSN 0969-6989, 2019 [Võrgumaterjal]. Available: https://doi.org/10.1016/j.jretconser.2019.05.025 . [Kasutatud 20 november 2022].

[8] Royi Benita, „Clean Node.js Architecture —With NestJs and TypeScript“, Better Programming, Jan. 2022 [Võrgumaterjal]. Available: https://betterprogramming.pub/clean-node-js-architecture-with-nestjs-and-typescript-34b9398d790f . [Kasutatud 20 november 2022].

[9] „Introduction to JSON Web Tokens“ [Võrgumaterjal]. Available: https://jwt.io/introduction . [Kasutatud 20 november 2022].

[10] Perea, P., Giner, P. (n.d.), „UX Design for Mobile Design apps that deliver impressive mobile experiences“, Packt Publishing, The Limited, 2017 [Võrgumaterjal]. Available: https://learning.oreilly.com/library/view/ux-design-for/9781787283428/ad9475dd-c5fa-4675-b75b-0c1eea1257ce.xhtml . [Kasutatud 20 november 2022].

[11] Krasimir H. , „MySQL vs PostgreSQL -- Choose the Right Database for Your Project,“ okta Developer, Jul. 2019 [Võrgumaterjal]. Available: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres . [Kasutatud 20 november 2022].

[12] „What Is Nest.js? A Look at the Lightweight JavaScript Framework“ [Võrgumaterjal]. Available: https://kinsta.com/knowledgebase/nestjs/#what-is-nestjs . [Kasutatud 20 november 2022]

[13] Ann M., „Why Use Nest.js“, DevCycle, Nov. 2021 [Võrgumaterjal]. Available: https://devcycle.com/blog/why-use-nest-js . [Kasutatud 20 november 2022].

[14] „What Is React Native? Is It Worth Using?“ [Võrgumaterjal]. Available: https://brainhub.eu/library/what-is-react-native . [Kasutatud 20 november 2022].

[15] „What is expo used for?“ [Võrgumaterjal]. Available: https://docs.expo.dev/introduction/faq/ . [Kasutatud 20 november 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[8]

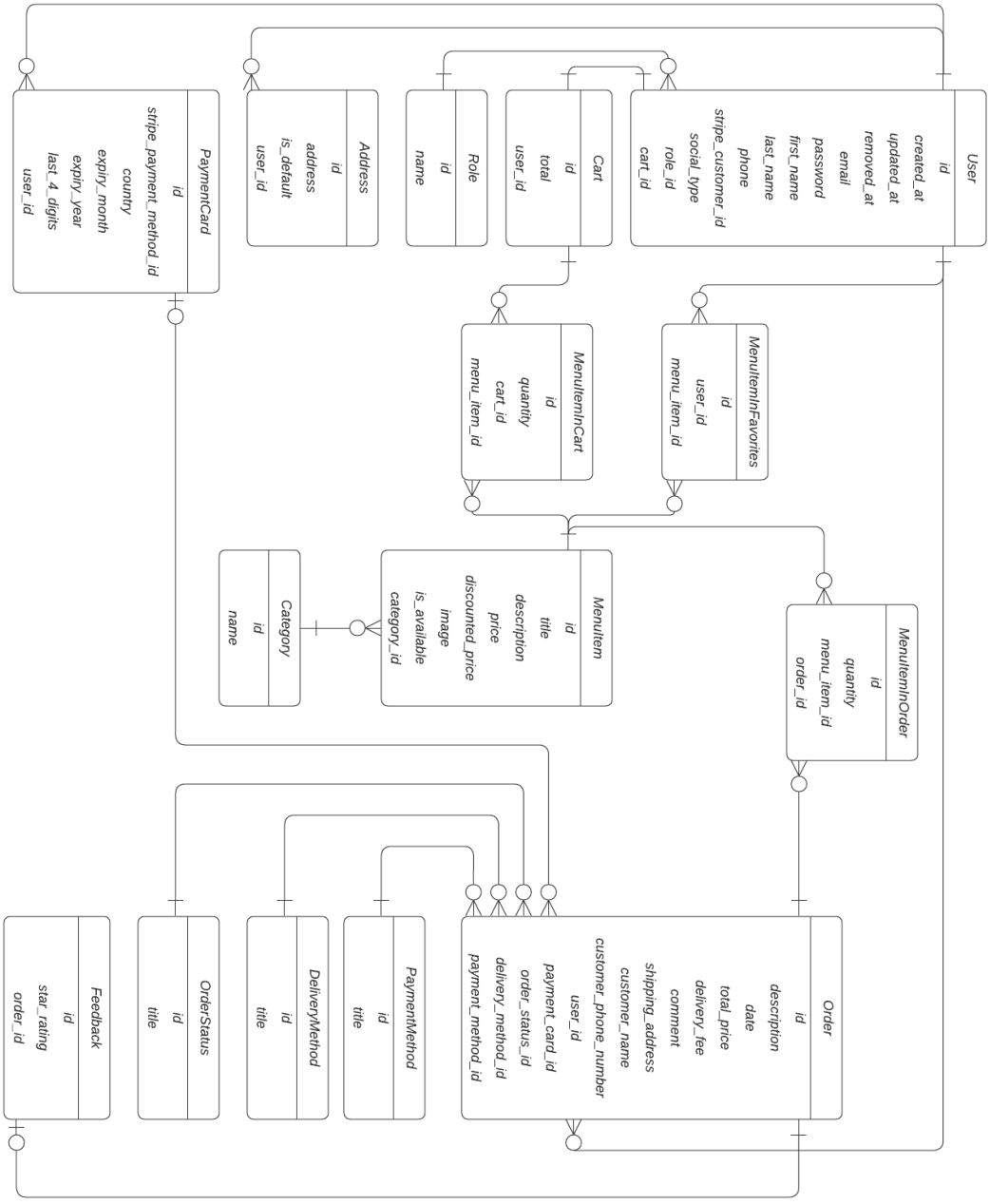
Mina, Ilja Ivanov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Restorani toidu tellimise mobiilirakenduse ja tagarakenduse arendamine", mille juhendaja on German Mumma
   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.

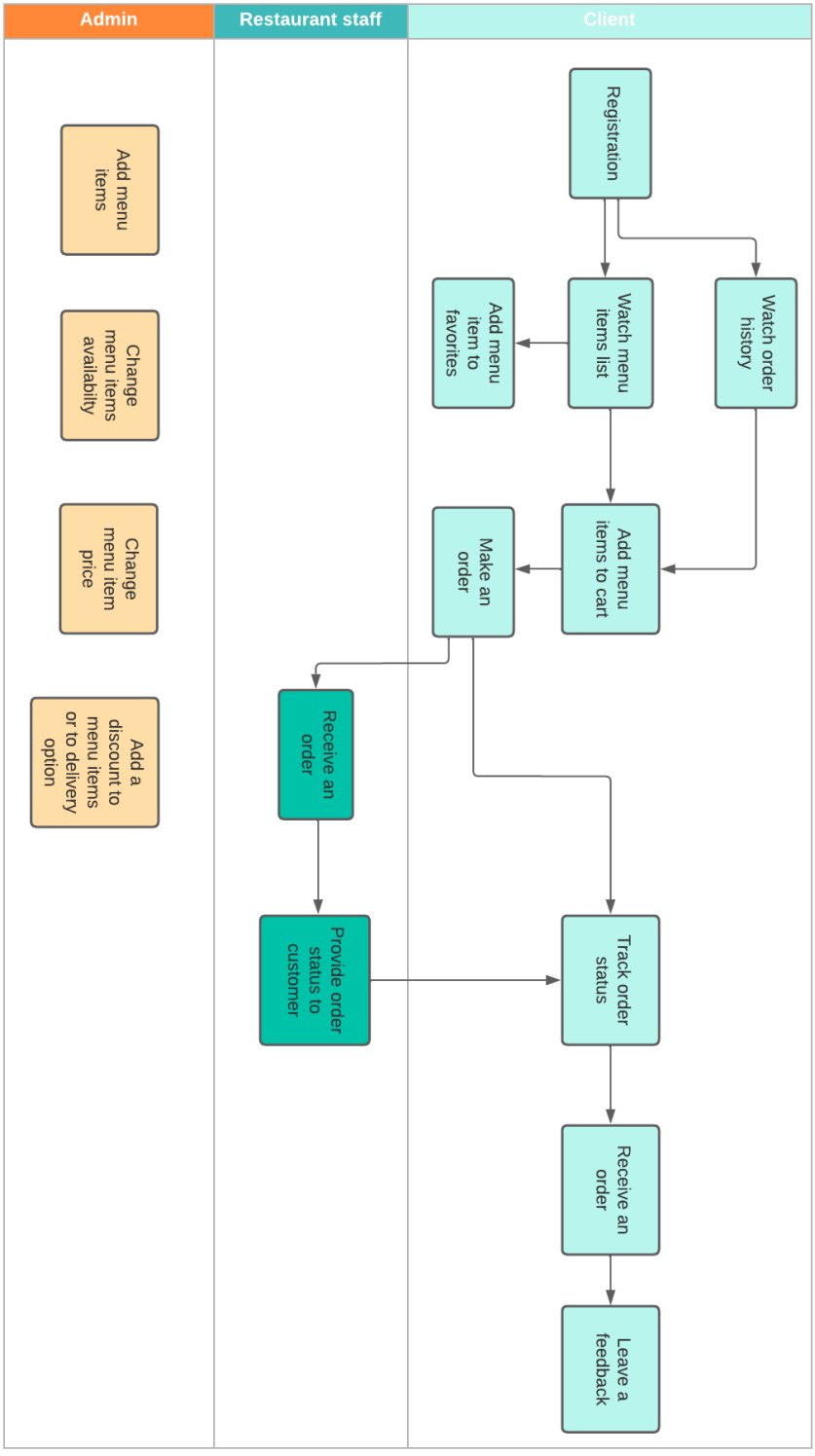
05.01.2023

8 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 – Andmebaasi skeem



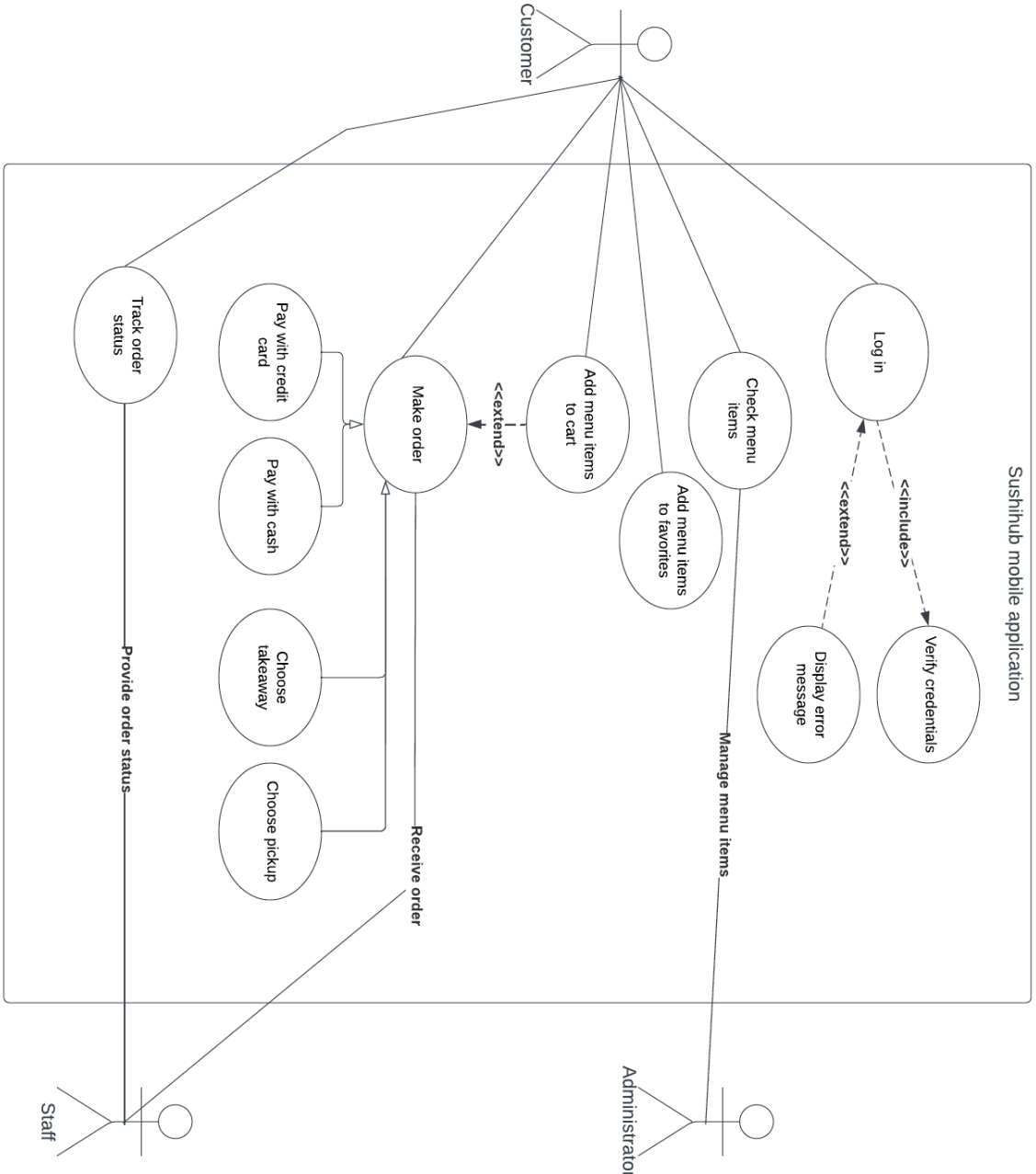
Lisa 3 – Äriprotsesside voog



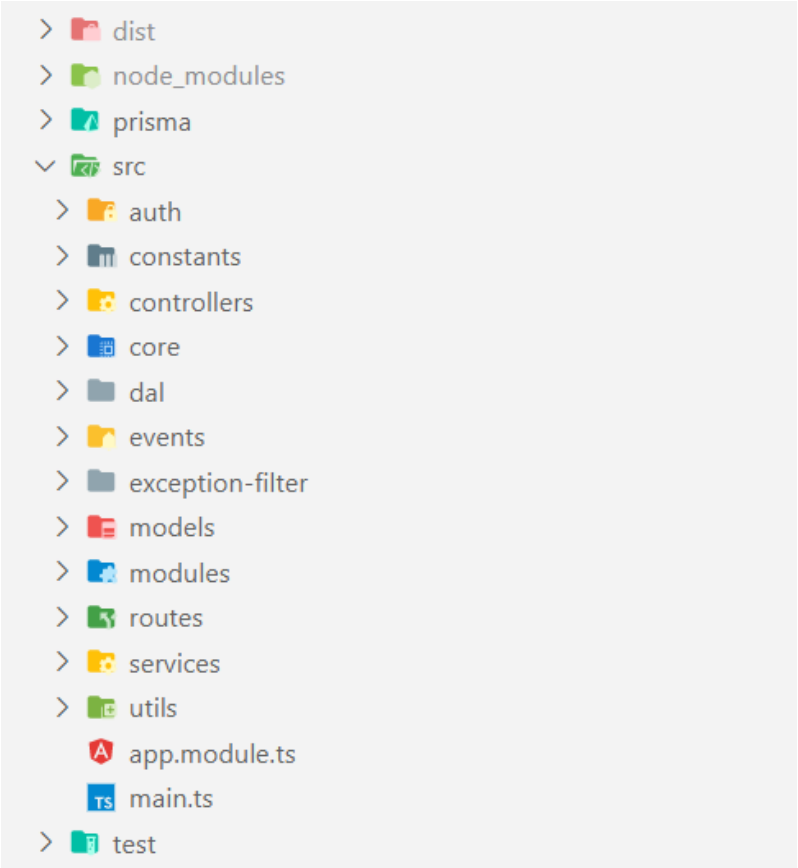
Lisa 4 – Kasutajalood

ID	Priority	As a <type of user>	I want to <perform some task>	so that I can <achieve some goal>
1	Low	kliendina	soovin luua enda kontot	selleks, et näha tellimuste ajalugu
2	High	kliendina	soovin näha restorani menüüd	selleks, et valida toitu, mida sooviksin tellida
3	High	kliendina	soovin, et oleks võimalus valida toidu kategooriat	selleks, et oleks lihtsam valida otsitavat toitu
4	Medium/Low	kliendina	soovin tellimuse vormistamise ajal automaatselt valida enda asukohta	selleks, et määrata täpset asukohta
5	Medium	kliendina	soovin jälgida oma tellimuse staatust	selleks, et näha millises etapis on minu tellimus
6	High	kliendina	soovin valida kohaletoimetamise viisi	selleks, et oleks võmalus mugava viisi valimiseks
7	High	kliendina	soovin valida maksmisemeetodit	selleks, et oleks võimalus mugavalt maksta
8	High	klienditeenindajana	soovin võtta tellimuste vastu	selleks, et kliendil oleks võmailus toitu tellida
9	Medium	klienditeenindajana	soovin muuta tellimuste staatust	selleks, et klient näeks, millises etapis on tema tellimus
10	Medium/High	adminina	soovin muuta toitude hindu	selleks, et tagada ettevõtte majanduslikku tasakaalu
11	Medum/High	adminina	soovin toitu menüüsse lisada või sealt eemaldada	selleks, et oleks võimalus uuendada restorani menüüd

Lisa 5 – Kasutusjuhtude diagramm



Lisa 6 – Tagarakenduse projekti struktuur



Lisa 7 – Mobiilirakenduse projekti struktuur