TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Diana Christine Tarro 194070IADB

# Ilusalongi aegade broneerimise veebirakenduse arendus

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: Diana Christine Tarro

23.04.2023

# Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua ilusalongile veebirakendus, mis võimaldab salongi klientidel iseseivalt ja mugavalt iluteenuste broneeringuid teha. Samuti võimaldab veebirakendus salongi iluteenindajal hallata oma töögraafikut ja iluteenuseid.

Töö teoreetilises osas kirjeldatakse lahendatavat probleemi ja olemasolevaid lahendusi ning pannakse paika uue lahenduse skoop. Samuti määratakse rakenduse nõuded ning analüüsitakse erinevaid tehnoloogiaid ja valitakse neist probleemi lahendamiseks kõige sobivamad. Lisaks kirjeldatakse veebirakenduse arhitektuuri ning kasutajakogemuse ja andmebaasi disaini.

Töö praktilises osas luuakse ja testitakse veebirakendust ning selle käigus selgitatakse erinevaid aspekte nii veebiteenuse kui ka klientrakenduse arendusel. Valminud veebirakendusele antakse hinnang ja pakutakse välja erinevaid võimalusi edasiarendusteks. Töö tulemuseks on kasutusvalmis veebirakendus, mis võimaldab ilusalongi teenustele aegu broneerida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki, 18 joonist, 3 tabelit.

# Abstract

## Development of a Web Application for Booking Beauty Salon Appointments

The aim of this thesis is to create a web application for a beauty salon, which allows salon customers to book beauty services independently and conveniently. The web application also allows the salon's beautician to manage their own work schedule and beauty services.

In the theoretical part of the thesis, the problem to be solved and the existing solutions are described, as well as the scope of the new solution is set. Also, the requirements of the application are defined, different technologies are analysed, and the most suitable ones are selected to solve the problem. In addition, the architecture of the web application as well as the design of the user experience and the database are described.

In the practical part of the thesis, a web application is created and tested, and in the process, various aspects of both web service and client application development are explained. The finalized web application is evaluated as well as different possibilities for further development are suggested. The result of the work is a ready-to-use web application that allows booking appointments for beauty salon services.

The thesis is in Estonian and contains 31 pages of text, 6 chapters, 18 figures, 3 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| ACID | *Atomicity, Consistency, Isolation, and Durability* – atomaarsus, konsistentsus, isoleeritus, püsivus |
| API | *Application Programming Interface* – rakendusliides |
| BBL | *Business Logic Layer* – äriloogikakiht |
| CSS | *Cascading Style Sheets* – kaskaadlaadistik |
| DAL | *Data Access Layer* – andmete juurdepääsu kiht |
| DTO | *Data Transfer Object* – andmeedastusobjekt |
| ERD | *Entity Relationship Diagram* – olemi-suhte diagramm |
| HTML | *HyperText Markup Language* – hüpertekst-märgistuskeel |
| JSON | *JavaScript Object Notation* – JavaScripti andmevahetusvorming |
| JWT | *JSON Web Token* – JSON-vormingus veebi turvamärk |
| LINQ | *Language-Integrated Query* – keelega integreeritud päring |
| MS | Microsoft |
| NoSQL | *non-SQL, not only SQL* – mitte-SQL, mitte ainult SQL |
| SQL | *Structured Query Language* – struktuurpäringukeel |
| UI | *User Interface* – kasutajaliides |
| URL | *Uniform Resource Locator* – internetiaadress |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Veebirakenduse kasutamine iluteenuste aegade broneerimiseks on praegusel ajal Eesti ilusalongides väga levinud, kuna sel viisil broneeringuid teha ning hallata on lihtne ja mugav nii kliendi kui ka salongi töötaja jaoks. Broneerimissüsteemi kasutamine teeb klientidele nähtavaks iluteenindaja töögraafikus olevad vabad ajad ning kliendid saavad endale ise sobival hetkel iluteenindaja töögraafikusse erinevate iluteenuste broneeringuid teha.

Tallinnas Nõmmel asuvas väikeses ühe töötajaga ilusalongis ei kasutata iluteenuste aegade broneerimiseks veebirakendust ning broneeringu tegemise protsess on ebaefektiivne ning aeganõudev – aja broneerimiseks peab klient vestlema salongi töötajaga ja teenuse aja sel viisil kokku leppima. Salongi töötaja peab pidevalt arvutama kuidas ta ajaliselt jõuab tegeleda samal päeval mitmete broneeringutega ning seejuures arvestama ka seda, kui kaua igale iluteenusele aega kulub (sealhulgas aeg ettevalmistuseks ja koristamiseks). Praegune broneeringu tegemise protsess vähendab töötaja aega, mida oleks võimalik kasutada iluteenuste teostamiseks ning ettevõtte kasumi suurendamiseks.

Käesoleva lõputöö eesmärk on luua veebirakendus ilusalongi teenuste aegade broneerimise protsessi lihtsustamiseks. Veebirakendus võimaldaks klientidel ise mugavalt iluteenuste broneeringuid teha ning salongi töötaja ei peaks enam broneeringute töögraafikusse sobitamisega tegelema ja tal oleks võimalik rohkem aega panustada erinevate iluteenuste teostamisele.

## 1.1 Metoodika

Käesolevas lõputöös kirjeldatakse esmalt lahendatavat probleemi. Seejärel vaadeldakse olemasolevaid lahendusi ja tuuakse esile nende puudused antud probleemi lahendamisel. Eelneva põhjal pakutakse välja uus lahendus ning pannakse paika selle skoop.

Loodavat lahendust analüüsitakse. Selle jaoks määratakse kõigepealt rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded ning kirjeldatakse lühidalt seda, kuidas antud nõuded saadi. Analüüsi käigus võrreldakse erinevaid tehnoloogiaid, valitakse välja kõige sobivamad antud probleemi lahendamiseks ning põhjendatakse valikuid. Analüüsitakse ka veebirakenduse arhitektuuri ja kasutajakogemuse disaini. Lisaks disainitakse rakenduse nõuetele vastav andmebaas.

Veebirakenduse arenduse käigus kirjeldatakse nii veebiteenuse kui ka klientrakenduse arendusprotsessi. Kui veebirakendus on valminud, siis viiakse läbi eksperiment, mille käigus testitakse veebirakenduse funktsionaalseid nõudeid ning kirjeldatakse testimise protsessi.

Lõputöö raames valminud rakendusele antakse hinnang ning pakutakse välja erinevaid võimalusi edasiarendusteks.

# 2 Probleemi ülevaade

Ilusalongis on teenuste broneeringute tegemine hetkel aeganõudev protsess, kuna uue aja broneerimine või olemasoleva broneeringu muutmine eeldab vestlust salongi töötajaga, kellel ei ole võimalik igal hetkel kliendile vastata (näiteks töövälisel ajal või parasjagu teise kliendiga tegeledes). On juhtunud, et samal ajal kui klient iluteenindaja vastust ootab, leiab ta teise salongi, saab sealt kiiremini oma soovitud iluteenusele aja ning läheb sinna. Ebaefektiivse aja broneerimise protsessi tõttu on salong kaotanud kliente konkurentidele.

Lisaks sellele raskendab praegune lahendus iluteenindaja tööd, sest ta peab ise oma töögraafikusse iluteenuste broneeringuid sobitama ning pidevalt arvutama, kuidas oma tööaega planeerida. See võib olla mõnikord keeruline, kuna samale päevale võivad iluteenuse broneeringut tahta mitu erinevat klienti. Broneeringuid töögraafikusse lisades tuleb arvestada, et iluteenused on erineva kestusega ning iluteenuste vahele peab jääma aeg koristamiseks ja järgmise teenuse ettevalmistamiseks. Selline arvutamine on aeganõudev ja veaohtlik. Iluteenuste töögraafikusse sobitamine vähendab iluteenindaja aega, mida oleks võimalik kasutada klientidele iluteenuste teostamiseks ning ilusalongi tulu kasvatamiseks.

## 2.1 Olemasolevad lahendused

Järgnevalt vaadeldakse mõningaid Tallinnas populaarseid ilusalongide broneerimissüsteeme ning tuuakse välja ka nende puudused antud probleemi lahendamisel.

### 2.1.1 SalonInfra

SalonInfra on Tallinna ilusalongides väga laialdaselt kasutuses olev tarkvara, mis võimaldab iluteenindajatel hallata töögraafikut ja näha kliendiajalugu ning klientidel teha iluteenuste broneeringuid. Paraku on tegu üsna kalli lahendusega – ühe töötajaga salongile on paketi kuutasu hind alates 35€ (+ käibemaks 20%) ning see pakett ei sisalda paljusid funktsionaalsusi, näiteks soodustuste ja kinkekaartide loomist ning broneeringute

e-maili teavitusi [1]. SalonInfra puudusteks on seega kõrge hind ning antud probleemi lahendamiseks kasulike funktsionaalsuste puudumine.

### 2.1.2 Booklux

Booklux on broneerimistarkvara, millel on ka oma mobiilirakendus. Tarkvara võimaldab klientidel teha iluteenindaja töögraafikusse broneeringuid ning iluteenindajal näha kliendiajalugu ja oma töögraafikut hallata. Iluteenindaja saab ka hallata oma iluteenuseid, kuid teenusele pole võimalik määrata ettevalmistus- ja koristusaega. Ühe töötajaga ilusalongile on Bookluxi kuutasu 11.99€ (+ käibemaks), kuid paraku ei sisalda see kõiki funktsionaalsusi. Näiteks soodustuste tegemise funktsionaalsuse eest tuleb veel lisaks tasuda igakuiselt alates 19€ [2]. Bookluxi puuduseks on see, et erinevate kasulike lisafunktsionaalsuste lisamisel võib broneerimistarkvara kuutasu kasvada üsna suureks. Lisaks on Bookluxi miinuseks ka see, et teenusele pole võimalik määrata ettevalmistus- ja koristusaega, kuid antud probleemi lahendamisel on see oluline.

### 2.1.3 SalonLife

SalonLife on ilusalongi platvorm, mis sisaldab salongikalendrit, ülevaadet iluteenindaja töögraafikust ja broneeringutest, kliendiajalugu ning statistikat. Lisaks on võimalik luua kinkekaarte ja seadistada broneeringute meeldetuletusi klientidele. SalonLife kuutasu ilusalongile on 17€ (+ käibemaks) ning see sisaldab kõiki pakutavaid funktsionaalsusi. Lõputöö kirjutamise hetkel saavad SalonLife platvormil kasutajakonto luua ainult salongi töötajad ning klientidel kasutaja loomise võimlaust pole [3]. Antud probleemi lahendamisel on broneerimissüsteemis kliendi kasutajakonto olemasolu oluline, kuna siis saaks klient näha ülevaadet oma broneeringutest ja tal oleks võimalik oma tulevasi broneeringuid ise mugavalt muuta. Lisaks oleksid kliendi andmed salvestatud ning tänu sellele oleks kliendil võimalik teha broneeringuid kiiremini, kuna broneeringut tehes poleks vajadust oma andmeid iga kord uuesti lisada. Seega on kliendi kasutajakonto puudumine SalonLife platvormi puuduseks.

## 2.2 Uue lahenduse skoop

Olemasolevatel lahendustel esineb puudusi ning seetõttu ei ole need kõige sobivamad antud lõputöös käsiteldava probleemi lahendamiseks.

Selle tõttu pakub lõputöö autor probleemi lahendamiseks välja uue lahenduse, milleks on luua veebirakendus, mis võimaldab salongi klientidel luua ise uusi broneeringuid ja vajadusel neid muuta ning iluteenindajal hallata oma töögraafikut, broneeringuid, kliente ja teenuseid. Rakendus teeb iluteenindaja eest ära vajalikud arvutused broneeringute töögraafikusse lisamisel ning arvestab ka iluteenuse ettevalmistamiseks ja hiljem koristamiseks kuluva ajaga, mille iluteenindaja on ise teenusele eelnevalt määranud. Kliendid saavad rakenduses teha oma kasutajakonto, kuid see pole kohustuslik, kuna iluteenuse aega on kliendil võimalik broneerida ka ilma kasutajata. Kasutajakontot omades on kliendil võimalik näha oma kõiki broneeringuid ning tulevasi broneeringuid lihtsasti muuta või tühistada.

Lahendus piiritleb vaid salongi iluteenuste aegade broneerimise protsessiga seotud tegevustega, kuid jätab võimaluse tulevikus lisada ilusalongi veebirakendusele ka teisi funktsionaalsusi, näiteks kinkekaartide loomine ja ostmine, statistika kuvamine, arvete koostamine ja saatmine.

Lõputöö kontekstis piirdutakse vaid veebirakendusega. Eelkõige seetõttu, et seda saab kasutada nii arvutis kui ka nutiseadmes. Veebirakendusele pääseb ligi brauseri kaudu ning veebirakendust ei pea eraldi alla laadima erinevalt näiteks mobiilirakendusest [4].

Lõputöö eesmärk on luua veebirakendus salongi iluteenuste aegade broneerimise protsessi kiirendamiseks ja lihtsustamiseks. Esialgu luuakse lahendus vaid ühele ilusalongile ning veebirakenduse loomisel lähtutakse peamiselt selle ettevõtte vajadustest.

# 3 Loodava veebirakenduse analüüs

Järgnevalt määratakse rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded ning analüüsitakse erinevaid tehnoloogiad ja valitakse neist sobivaimad. Lisaks analüüsitakse veebirakenduse arhitektuuri ja kasutajaliidese disaini ning disainitakse rakenduse nõuetele vastav andmebaas.

## 3.1 Nõuete määramine

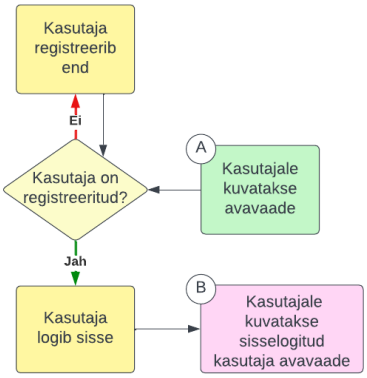
Veebirakenduse nõuete loomiseks kaardistati kõigepealt ilusalongi tegevused ja vajadused. Seejärel vesteldi salongiga, saadi tagasisidet ning selle põhjal koostati veebirakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuded põhinevad kasutajalugudel, mille rollideks on salongi töötaja ja salongi klient. Kõikide nõuete illustreerimiseks on loodud ka plokkskeem (Lisa 2). Plokkskeemil kujutab kollane värv sisselogimata kasutaja tegevusi, roosa värv sisselogitud kasutaja toiminguid ning lilla värv ainult sisselogitud iluteenindajale kättesaadavaid tegevusi.

### 3.1.1 Funktsionaalsed nõuded

Salongi töötaja- ja kliendi-põhised funktsionaalsed nõuded:

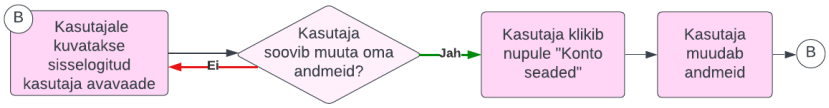
- „Salongi töötajana/kliendina soovin registreerida kasutajaks“ (Joonis 1). Rakendust avades kuvatakse kasutajale avavaade. Kasutaja navigeerib end kasutajaks registreerimise vaatesse, kus kuvatakse registreerimisvorm. Kasutaja täidab väljad ja vajutab nupule „Registreeri“ ning seejärel on uus kasutaja registreeritud.

- „Salongi töötajana/kliendina soovin logida sisse ja välja“ (Joonis 1). Kasutajale kuvatakse avavaade ning kasutaja vajutab nupule „Logi sisse“. Kasutaja täidab oma kasutajanime ja salasõna väljad ning vajutab nupule „Sisene“. Kui andmed olid õiged, siis logitakse kasutaja sisse ning kuvatakse sisselogitud kasutaja avavaade. Välja logimiseks tuleb kasutajal vajutada nuppu „Logi välja“.




Joonis 1. Kasutaja registreerimise ja sisselogimise plokkskeem.

- „Salongi töötajana/kliendina soovin muuta oma andmeid“ (Joonis 2). Sisselogitud kasutaja vajutab nupule „Konto seaded“ ning talle kuvatakse tema kasutajakonto andmed, mida tal on seejärel võimalik muuta.

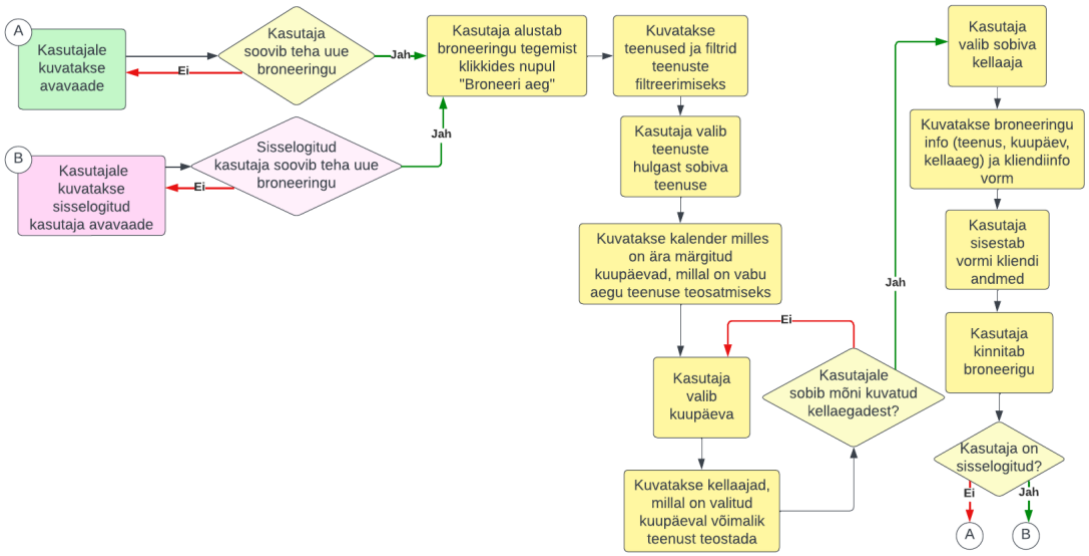



Joonis 2. Kasutaja andmete muutmise plokkskeem.

- „Salongi töötajana/kliendina soovin teha uue broneeringu veebirakendusse sisse logituna“ (Joonis 3). Sisselogitud kasutaja vajutab avavaates nupule „Broneeri aeg“. Kasutaja valib kuvatud iluteenustest sobiva teenuse. Seejärel kuvatakse kasutajale kalender, millest kasutaja valib broneeringu kuupäeva. Kui kuupäev on valitud, siis valib kasutaja broneeringu kellaja. Seejärel kuvatakse sisselogitud kasutajale tema andmetega täidetud kliendiinfo vorm, mida on kasutajal soovi korral võimalik muuta. Kui kasutaja vajutab nuppu „Kinnita broneering“, siis broneering kinnitatakse.

- „Salongi töötajana/kliendina soovin teha uue broneeringu veebirakendusse sisse logimata“ (Joonis 3). Sisselogimata kasutaja broneeringu tegemine käib enamjaolt samamoodi nagu sisselogitud kasutajal. Ainus erinevus on kliendiinfo vormi täitmine, kuna sisselogimata kasutajal pole see eelnevalt täidetud ning ta peab vormi ise täitma.

- „Salongi töötajana/kliendina soovin broneeringut tehes näha teenuse hinda ja ajakulu“ (Joonis 3). Broneeringut tehes saab iga teenuse hinda ja ajakulu näha iluteenuse valimise vaates.

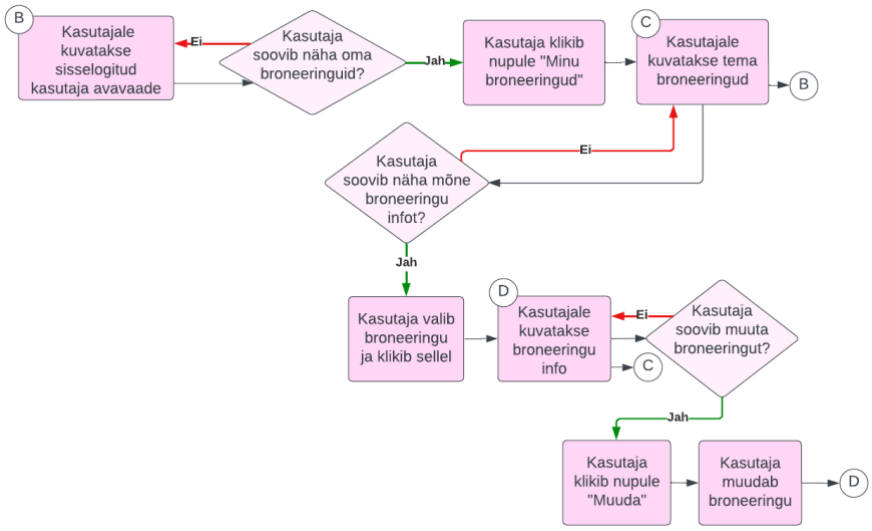



Joonis 3. Broneeringu loomise plokkskeem.

- „Salongi töötajana/kliendina soovin näha oma kõiki broneeringuid“ (Joonis 4). Oma kõikide broneeringute nägemiseks vajutab sisselogitud kasutaja nupule „Minu broneeringud“ ning seejärel kuvatakse kasutajale kõik tema broneeringud.

- „Salongi töötajana/kliendina soovin muuta ja kustutada oma tulevasi broneeringuid“ (Joonis 4). Broneeringute muutmiseks või kustutamiseks valib sisselogitud kasutaja oma kõikide broneeringute hulgast ühe broneeringu ning seejärel kuvatakse talle valitud broneeringu info. Vajutades nuppu „Muuda“ on võimalik broneeringut muuta või kustutada.

- „Salongi töötajana/kliendina soovin rakenduses näha pilte tehtud töödest“ (Joonis 4). Juba toimunud broneeringu info vaates on võimalik näha selle broneeringu käigus tehtud tööst pilti kui iluteenindaja on selle sinna lisanud.

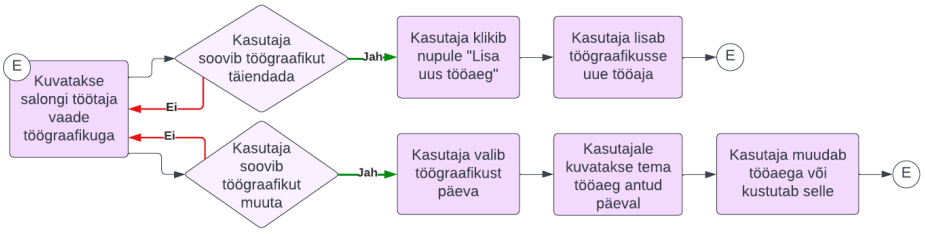



Joonis 4. Kasutaja broneeringute plokkskeem.

Salongi töötaja-põhised funktsionaalsed nõuded:

- „Salongi töötajana soovin lisada pilte tehtud töödest“ (Joonis 4). Sisselogitud salongi töötajal on juba toimunud broneeringu muutmise vaates võimalik lisada selle broneeringu käigus tehtud tööst pilt.

- „Salongi töötajana soovin hallata oma töögraafikut“ (Joonis 5). Sisselogitud salongi töötajale kuvatakse tema avavaade koos töögraafikuga. Kui kasutaja soovib töögraafikusse tööaega lisada, siis vajutab ta nupule „Lisa uus tööaeg“ ning lisab uue tööaja. Kui kasutaja soovib olemasolevat töögraafikut muuta, siis valib ta oma töögraafikust päeva ning talle kuvatakse tema tööaeg valitud päeval. Seejärel on kasutajal võimalik tööaega muuta või kustutada.



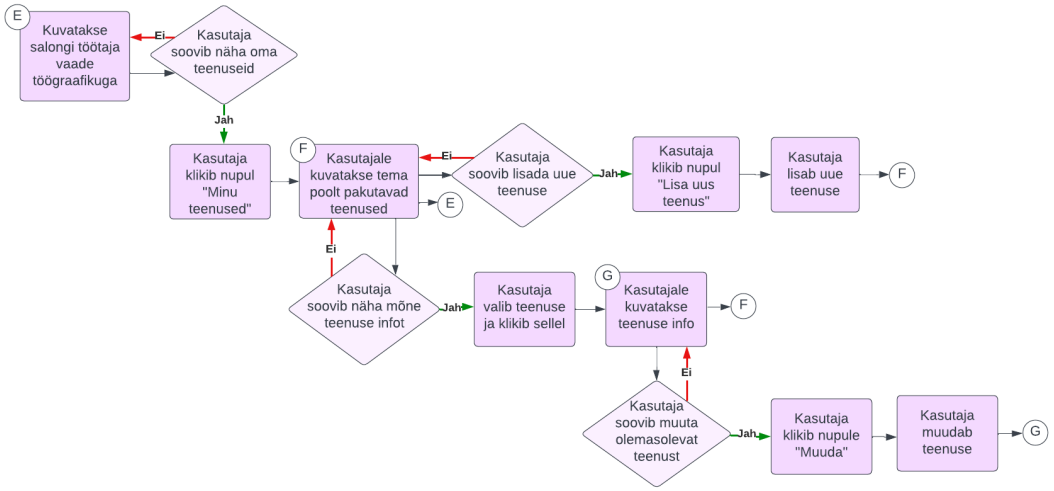

Joonis 5. Salongi töötaja töögraafiku plokkskeem.

- „Salongi töötajana soovin lisada, muuta ja kustutada oma teenuseid“ (Joonis 6). Sisselogitud salongi töötaja vajutab nupule „Minu teenused“ ning talle kuvatakse

kõik tema iluteenused. Kui kasutaja soovib lisada uue teenuse, siis vajutab ta nupule „Lisa uus teenus“ ning lisab uue teenuse. Kui kasutaja soovib olemasolevat teenust muuta, siis valib ta oma teenuste hulgast ühe teenuse ning talle kuvatakse teenuse info. Seejärel on kasutajal võimalik teenust muuta või kustutada.

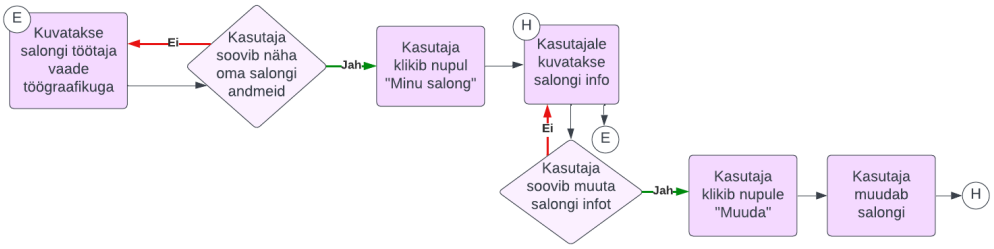
- „Salongi töötajana soovin muuta oma teenuste hindu“ (Joonis 6). Sisselogitud salongi töötajal on teenuse hinda võimalik muuta teenuse muutmise vaates.

- „Salongi töötajana soovin lisada, muuta ja kustutada oma teenuste soodustusi“ (Joonis 6). Sisselogitud salongi töötajal on võimalik teenusele soodustusi lisada ning neid muuta ja kustutada samuti teenuse muutmise vaates.




Joonis 6. Salongi töötaja teenuste plokkskeem.

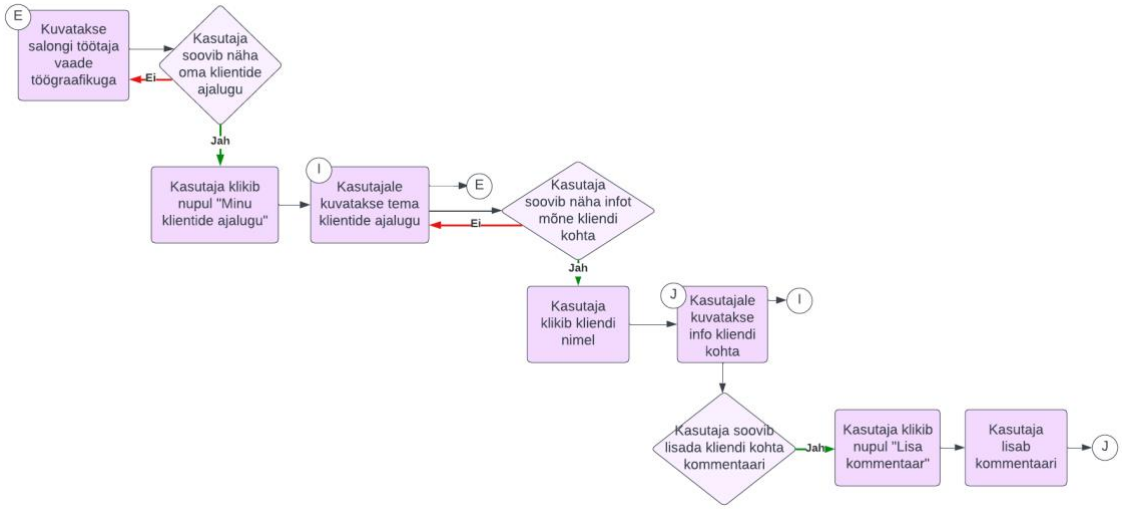
- „Salongi töötajana soovin näha ja muuta salongi infot“ (Joonis 7). Sisselogitud salongi töötaja vajutab nupule „Minu salong“ ning talle kuvatakse tema salongi info. Kui kasutaja soovib salongi infot muuta, siis vajutab ta nupule „Muuda“ ning seejärel on tal võimalik salongi infot muuta.




Joonis 7. Salongi töötaja salongi info plokkskeem.

- „Salongi töötajana soovin näha kliendi ajalugu minu osutatud iluteenustest“ (Joonis 8). Sisselogitud salongi töötaja vajutab nupule „Minu klientide ajalugu“ ning talle kuvatakse tema klientide ajalugu. Kliendi info nägemiseks valib kasutaja mõne kliendi ning talle kuvatakse valitud kliendi info.

- „Salongi töötajana soovin lisada klientide kohta kommentaare“ (Joonis 8). Sisselogitud salongi töötajal on võimalik kliendi info vaates vajutada nupule „Lisa kommentaar“ ning seejärel lisada kliendi kohta kommentaar.




Joonis 8. Salongi töötaja klientide plokkskeem.

### 3.1.2 Mittefunktsionaalsed nõuded

Kõigi kasutajarollide mittefunktsionaalsed nõuded:

- Soovin, et minu andmeid oleksid turvalised.

- Soovin, et veebirakendus on eestikeelne.

- Soovin, et veebirakenduse kasutamine oleks võimalikult intuitiivne ja lihtne.
- Soovin veebirakendust kasutada nutiseadmes ja arvutis.

## 3.2 Tehnoloogiate valik

Käesolevas lõputöös valmiva veebirakenduse tehnoloogiate valikul analüüsitakse ainult kaasaegseid tehnoloogiaid, millel on suur kasutajaskond. Tehnoloogiate valikul on arvestatud tehnoloogia sobivusega antud probleemi lahendamiseks ning lõputöö autori kogemusega.

### 3.2.1 Andmebaasisüsteem

Andmebaas on veebirakenduse väga oluline osa, kuna andmebaasis hoitakse struktureeritud kujul andmeid ning andmebaasi kasutatakse andmete efektiivseks haldamiseks. Andmebaasid saab jagada kahte kategooriasse – relatsioonilised andmebaasid (SQL) ja mitterelatsioonilised andmebaasid (NoSQL) [5].

Relatsioonilises andmebaasis on andmed struktureeritud, kuna andmeid hoitakse tabelites ja andmetabelite vahel on seosed. Relatsioonilisi andmebaase nimetatakse ka SQL andmebaasideks, sest need kasutavad SQL päringukeelt andmebaasi päringute teostamiseks ehk andmete pärimiseks, loomiseks, muutmiseks ja kustutamiseks [6]. Relatsioonilise andmebaasi heaks omaduseks on ACID (*Atomicity, Consistency, Isolation, and Durability*) standard, mis tagab andmebaasi transaktsioonide usaldusväärsuse, kuna see aitab vältida andmekadu ning mitteterviklikke andmeid. Selle üldine põhimõte on see, et kui üks muudatus ebaõnnestub, siis ebaõnnestub kogu transaktsioon ja andmebaas jääb olekusse, milles see oli enne transaktsiooni sooritamist [7]. Relatsioonilised andmebaasid sobivad tavaliselt hästi kui andmete hulk on väikese või keskmise suurusega [6].

Mitterelatsioonilised andmebaasid on paindlikumad ja vähem struktureeritud. Need on sellised andmebaasid, mis ei kasuta seostega andmetabeleid nagu relatsioonilised andmebaasid ning võivad kasutada SQL-i asemel mõnda teist päringukeelt. Mitterelatsioonilisi andmebaase on erinevaid, näiteks dokumentandmebaas, võti-väärtus andmebaas, graafandmebaas. Mitterelatsioonilised andmebaasid sobivad tavaliselt paremini suurte andmekogumitega töötamiseks [6].

Lõputöö autor on oma lahenduse jaoks valinud relatsioonilise andmebaasi, kuna see võimaldab loodava rakenduse jaoks vajalikke andmeid struktureerida tabelite ja nendevaheliste seostega ning kasutada SQL päringukeelt andmebaasi päringute teostamiseks. Kasulik on ka ACID standardi olemasolu, sest see kindlustab andmebaasi transaktsioonide usaldusväärsuse. Lisaks ei hakka loodav rakendus töötama väga suure andmete hulgaga, vaid pigem väikese kuni keskmise suurusega andmekogumitega.

Aastal 2022 Stack Overflow poolt korraldatud uuringust selgub, et kõige populaarsem andmebaasisüsteem aastal 2022 professionaalsete arendajate hulgas on relatsiooniline andmebaas PostgreSQL. Sellele järgnevad MySQL, SQLite ja Microsoft SQL Server, mis on samuti relatsioonilised andmebaasid [8].

PostgreSQL on avatud lähtekoodiga tasuta andmebaasisüsteem. PostgreSQL-i eeliseks on paljude lisafunktsioonide olemasolu. Üheks lisafunktsiooniks on näiteks suur andmetüüpide valik, mis sisaldab ka JSON-i andmetüüpi. PostgreSQL andmebaasisüsteemi on hinnatud kõige rohkem ACID standardit järgivaks andmebaasisüsteemiks [9].

MySQL on avatud lähtekoodiga tasuta andmebaasisüsteem. MySQL-i üheks tugevuseks on kiirus ning tegu on ühe väga populaarse ja laialdaselt kasutuses oleva andmebaasisüsteemiga. Kahjuks on alates MySQL-i omandamisest Oracle poolt on andmebaasisüsteemi areng aeglustunud [10].

SQLite on avatud lähtekoodiga tasuta andmebaasisüsteem. SQLite eeliseks on selle lihtne ülesseadistamine, kuna SQLite on serverita failipõhine andmebaasisüsteem. SQLite sobib hästi kasutamiseks testimisel ja mobiilirakendustes [10].

Microsoft SQL Server ehk MS SQL on andmebaasisüsteem, mis on arendatud Microsofti poolt. MS SQL kasutab päringute teostamiseks Transact-SQL (T-SQL) päringukeelt, mis on SQL-i laiendus. Kahjuks on MS SQL tasuline andmebaasisüsteem [11].

Andmebaasisüsteemiks on lõputöö autor valinud PostgreSQL andmebaasisüsteemi, kuna see on tasuta ning sellel on palju erinevaid ja kasulikke lisafunktsioone. Kindlasti tuleb kasuks ka PostgreSQL-i ACID standardi kõrge hinnang. Lisaks on autoril selle andmebaasisüsteemiga palju kogemust.

### 3.2.2 Teenuspoolse raamistiku ja programmeerimiskeele valik

Veebiteenuse ehk tagarakenduse raamistikud on tänapäeva veebiarendusemaailma väga olulised osad. Tagarakenduse raamistik aitab kiirendada ja automatiseerida veebiarendusprotsessi [12]. Raamistik pakub tööriistade komplekti, teeke ja standardset viisi veebirakenduse teenusepoole loomiseks ja haldamiseks, sealhulgas veebiserverit, andmebaasi, API-t ja teisi funktsioone nagu turvalisus, seansihaldus ja andmebaasi interaktsioon [13].

Tuntuimad tagarakenduse raamistikud [12], millega on autoril varasem kogemus on ASP.NET Core, Django ja Spring Boot.

ASP.NET Core on avatud lähtekoodiga, mitmeplatvormiline ja suure jõudlusega raamistik, mis on loodud Microsofti poolt. ASP.NET Core raamistik sobib hästi veebiteenuste ja veebirakenduste loomiseks [14]. ASP.NET Core raamistiku heaks küljeks on turvalisus, kuna see raamistik pakub laia valikut lihtsasti kasutatavaid mehhanisme autentimiseks, autoriseerimiseks, andmekaitseks ja rünnakute ennetamiseks. Selle raamistiku programmeerimiskeeleks on C#, mis on kaasaegne, objektorienteeritud ning tugevalt tüübitud programmeerimiskeel. Lisaks on C# programmeerimiskeelel ka väga hea asünkroonsuse tugi [15] ning C#-s on võimalik kasutada LINQ (*Language-Integrated Query*) tehnoloogiat, mis lihtsustab andmete töötlemist [16].

Django on avatud lähtekoodiga veebiarenduse raamistik, mis on disainitud selliselt, et sellega oleks võimalik kiirelt rakendusi luua. Django raamistikus on oluline turvalisus ning raamistik aitab arendajatel vältida paljusid levinumaid turvavigu. Django raamistiku programmeerimiskeeleks on Python [17]. Python on objektorienteeritud ja dünaamiliselt tüübitud programmeerimiskeel, mida on kerge õppida tänu oma lihtsale süntaksile [18].

Spring Boot on avatud lähtekoodiga veebiarenduse raamistik [12], mis kasutab endas Spring raamistikku. Veebiteenuse arendamist on Spring Boot raamistikus lihtne alustada, kuna tavaliselt on vaja selles raamistikus veebirakendust minimaalsest konfigureerida [19]. Spring Boot raamistikku on võimalik lisada ka Spring Security [20], mis on raamistik, mis pakub autentimist, autoriseerimist ja kaitset laialt levinud rünnakute vastu [21]. Spring Boot raamistiku programmeerimiskeeleks on Java [12]. Java on objektorienteeritud, tugevalt tüübitud ning väga populaarne programmeerimiskeel [22].

Järgnevalt on tabelites välja toodud autori kogemus eelnevalt mainitud teenusepoolsete raamistikega (Tabel 1) ja programmeerimiskeeltega (Tabel 2).

Tabel 1. Autori kogemus teenusepoolsete raamistikega.

| **Teenusepoolne raamistik** | **Autori kogemus** |
|---|---|
| ASP.NET Core | Hea |
| Django | Väike |
| Spring Boot | Keskmine |

Tabel 2. Programmeerimiskeelte võrdlus.

| **Programmeerimiskeel** | **Autori kogemus** | **Kogukonna suurus** |
|---|---|---|
| C# | Hea | 10 miljonit [23] |
| Python | Keskmine | 15,7 miljonit [23] |
| Java | Hea | 14 miljonit [23] |

Populaarsed tagarakenduse raamistikud on näiteks ka Laravel, Ruby on Rails, Express.js ja Flask [12], kuid lõputöö autoril puudub lõputöö kirjutamise hetkel nende raamistikega varasem kogemus. Uue raamistiku õppimine on aeganõudev ning seetõttu need variandid loodava rakenduse jaoks ei sobi.

ASP.NET Core, Django ja Spring Boot sobiksid kõik loodava veebirakenduse teenusepoole loomiseks, kuid lõputöö autor on oma lahenduse jaoks valinud ASP.NET Core raamistiku, mis kasutab C# programmeerimiskeelt. Valik on tehtud peamiselt lõputöö autori kogemuse põhjal – autoril on tagarakenduse raamistikest kõige rohkem kogemust ASP.NET Core raamistikuga ning programmeerimiskeeltest C#-iga. Autoril on hea kogemus ka Java programmeerimiskeelega, aga kuna Spring Boot raamistikuga on kogemust vähem kui ASP.NET Core raamistikuga, siis ASP.NET Core ja C# on antud olukorras paremad valikud. Lisaks tuleb selle valiku puhul kasuks ka C# programmeerimiskeeles kasutusel olev LINQ ja C#-i väga hea asünkroonsuse tugi.

### 3.2.3 Kliendipoolne tehnoloogia

Hea kasutajakogemus on veebirakenduse kasutajaliideses ülimalt oluline. Selle saavutamiseks on mõistlik kasutada mõnda klientrakenduse ehk esirakenduse

raamistikku, mis kiirendab ja lihtsustab kasutajaliidese arendamist erinevatele seadmetele (arvuti, nutitelefon jne) [24].

Esirakenduse raamistikud, millega on lõputöö autoril varasem kogemus on Aurelia.js, Vue.js ning React.js.

Aurelia.js on JavaScripti raamistik, mis on kergesti õpitav [25]. Raamistik on mõeldud nii väiksema kui ka suurema SPA (*Single-Page Application*) ehk üheleherakenduse arendamiseks [26]. Aurelia suureks miinuseks on selle vähene populaarsus. Stack Overflow on korraldanud aastal 2022 uuringu ning järjestanud ära selle põhjal 25 kõige populaarsemat veebiraamistikku ja veebitehnoloogiat ning Aurelia raamistiku nende hulgast ei leia [27].

Vue.js on populaarne [27] ning pidevalt arenev JavaScripti raamistik. See on lihtsasti arusaadav, tõhus ning mitmekülgne raamistik kasutajaliideste loomiseks [28]. Vue üks peamisi eeliseid on see, et ühe komponendi HTML, CSS ja JavaScript on kõik ühes failis ning seetõttu on see raamistik ka kergesti õpitav [29].

React.js on JavaScripti raamistik, mis on populaarsem kui eelnevalt mainitud raamistikud [27]. React raamistik on arendatud Facebooki poolt. React-i heaks küljeks see, et raamistik on komponendipõhine ning erinevaid komponente saab taaskasutada, mis säästab aega [24]. React raamistikus saab kasutada JSX-i (*JavaScript XML*), mis on JavaScripti süntaksilaiend [30].

Järgnevalt on tabelis välja toodud autori kogemus eelnevalt mainitud klientrakenduse raamistikega (Tabel 3).

Tabel 3. Autori kogemus klientrakenduse raamistikega.

| **Klientrakenduse raamistik** | **Autori kogemus** |
|---|---|
| Aurelia.js | Keskmine |
| Vue.js | Hea |
| React.js | Väike |

Lõputöö autor on oma lahenduse klientrakenduse raamistikuks valinud Vue.js raamistiku, kuna antud raamistik on populaarne, tõhus ja lihtsasti arusaadav ning autoril on selle raamistikuga kõige rohkem kogemusi. Populaarne klientrakenduse raamistik on ka React,

kuid kahjuks on autori kogemus selle raamistikuga väga väike ning piiratud aja tõttu see variant loodava veebirakenduse kliendipooleks ei sobi.

### 3.2.4 Versioonihalduskeskkonna valik

Versioonihalduskeskkonna valimiseks vaadeldakse kolme populaarset keskkonda: GitHub, GitLab ja Bitbucket [31].

GitHub on kõige kõrgemalt hinnatud ja lihtsasti kasutatav versioonihalduskeskkond [31] ning antud lõputöö kirjutamise hetkel Moz'i populaarseimate veebilehtede nimekirjas 28. kohal [32]. GitHub asutati aastal 2008 ja alates aastast 2018 kuulub see Microsoftile [33]. GitHub pakub mitmeid erinevaid lisafunktsioone, näiteks GitHub Copilot, mis aitab arendajal kiiremini programmeerida. GitHub-is on võimalik kasutada tasuta versiooni [34].

GitLab on ka kõrgelt hinnatud versioonihalduskeskkond [31]. GitLab on antud lõputöö kirjutamise hetkel Moz'i kõige populaarsemate veebilehtede nimekirjas 441. kohal [32]. GitLab-is on olemas nii tasuta kui ka tasulised versioonid ning GitLab-i esialgne versioon loodi aastal 2011 [35].

Bitbucket on samuti kõrgelt hinnatud versioonihalduskeskkond [31], kuid see ei ole nii populaarne kui eelnevalt nimetatud versioonihalduskeskkonnad [32]. Bitbucket asutati aastal 2008 ning alates aastast 2010 kuulub see Atlassianile [36]. Versioonihalduskeskkonnas Bitbucket on samuti võimalik kasutada tasuta versiooni [37].

Versioonihalduskeskkonnaks on lõputöö autor valinud GitHub keskkonna, kuna see on väga populaarne ning seda on lihtne kasutada. Lisaks on autoril selle versioonihalduskeskkonnaga palju kogemust.

## 3.3 Veebirakenduse arhitektuur

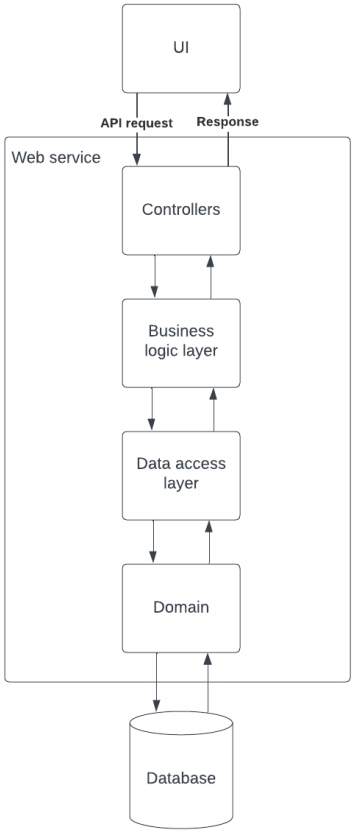
Hea tarkvara arhitektuur tagab tarkvara pika eluea ning selle, et tarkvara on võimalikult lihtsasti muudetav [38].

Veebirakenduse arhitektuuri loomisel on lähtutud puhta arhitektuuri (*The Clean Architecture*) ideedest. Puhas arhitektuur on selline tarkvara arhitektuur, kus on erinevad tarkvara osad jagatud loogilistesse kihtidesse. Selle eeliseks on see, et rakendus on

sõltumatu näiteks andmebaasist ja kasutajaliidesest, mis tähendab, et vajaduse korral on neid võimalik välja vahetada ilma äriloogikat muutmata. Lisaks lihtsustab puhas arhitektuur veebirakenduse testimist, kuna rakenduse osi on võimalik testida eraldi, näiteks on võimalik rakenduse äriloogikat testida ilma andmebaasi ja kasutajaliideseta [38].

Loodava veebirakenduse arhitektuur (Joonis 9) koosneb järgmistest kihtidest:

- Kasutajaliides (UI) – klientrakendus ehk see osa veebirakendusest, mida kasutaja näeb ja kasutab rakendusega suhtlemiseks.
- Kontrollerid (*Controllers*) – API kontrollerid, mis tegelevad API päringutega ja koostavad päringutele vastuseid.
- Äriloogikakiht (*Business logic layer*) – veebirakenduse äriloogika, mis on jagatud teenustesse.
- Andmete juurdepääsu kiht (*Data Access layer*) – veebirakenduse andmetele juurdepääs ja repositooriumid.
- Domeenikiht (*Domain*) – domeeniobjekte väljendavad klassid.
- Andmebaas (*Database*) – veebirakenduse andmebaas, kus hoitakse rakenduse andmeid.




Joonis 9. Veebirakenduse arhitektuur.

## 3.4 Veebirakenduse disain

Järgnevalt selgitatakse probleemi lahendamiseks loodava veebirakenduse kasutajakogemuse ja andmebaasi disaini. Nii kasutajakogemuse kui ka andmebaasi disaini loomiseks kasutati Lucidchart tööriista.

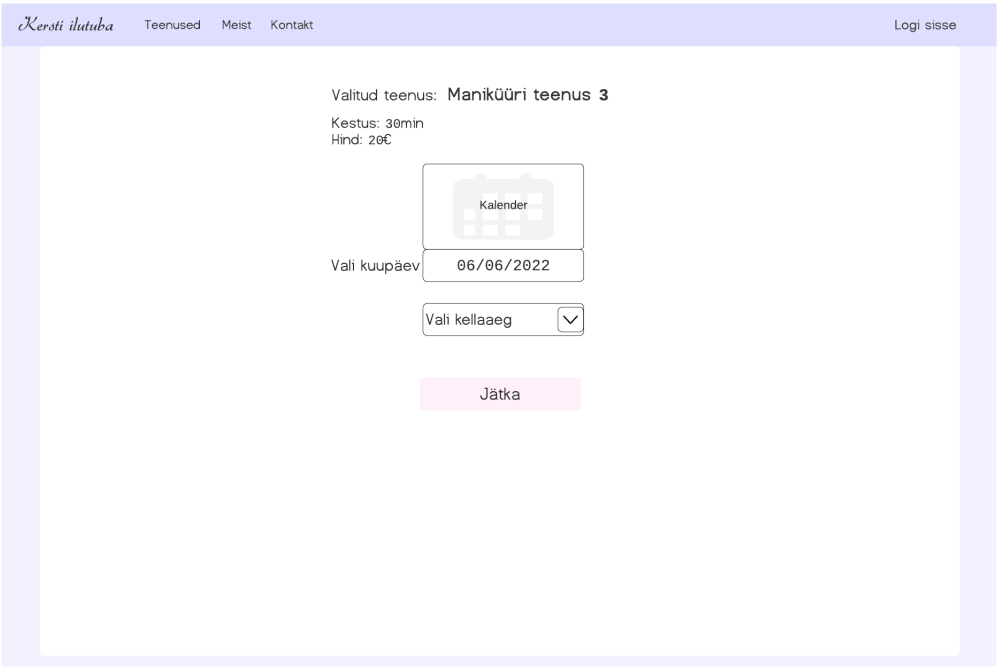
### 3.4.1 Kasutajakogemuse disain

Broneeringu loomise vaated on veebirakenduse kasutajakogemuse disaini kõige olulisem osa, kuna broneeringu loomine on veebirakenduse kõige olulisem funktsionaalsus.

Broneeringu loomiseks vajutab kasutaja avalehel nupule „Broneeri aeg“ ning seejärel kuvatakse talle broneeringu iluteenuse valiku vaade (Joonis 10). Vaate disainimisel on eeskujuks võetud ilusalongi hinnakiri. Salongi hinnakirjas on nii meeste kui ka naiste teenused, seega on iluteenuse valiku vaates teenuseid võimalik filtreerida teenuse sihtrühma (naised või mehed) järgi. Lisaks on salongi hinnakirjas teenused jagatud kategooriatesse (näiteks kulmud, maniküür). Iluteenuse valiku vaates on samuti kuvatud teenused kategooriates. Teenuseid ja teenuste kategooriaid saab iluteenindaja ise luua ning muuta.

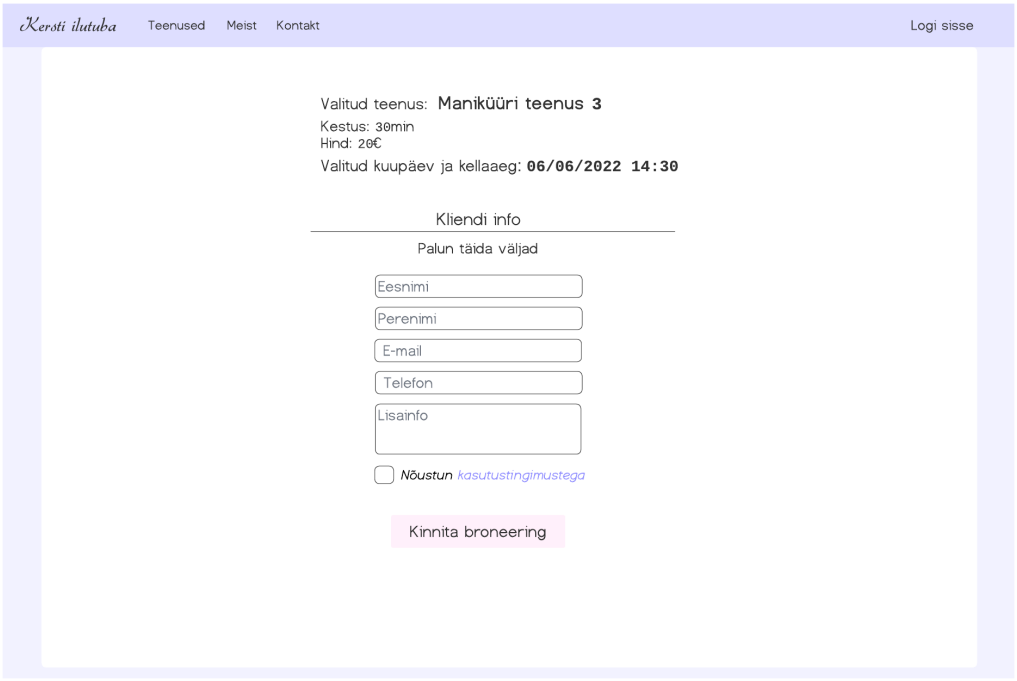

Joonis 10. Broneeringu iluteenuse valiku vaade.

Kui kasutaja on valinud sobiva iluteenuse, siis kuvatakse broneeringu aja valimise vaade (Joonis 11). Selles vaates kuvatakse kalender, kust kasutaja saab valida teenuse toimumiseks sobiva päeva. Kalendris on võimalik kasutajal valida vaid neid päevi, millel on valitud teenuse jaoks piisavalt vaba aega. Kui kuupäev on valitud, siis saab kasutaja valida ka omale sobiva kellaaja. Kui kasutajale ükski antud kellaegadest ei sobi, siis on võimalik valida uus kuupäev.



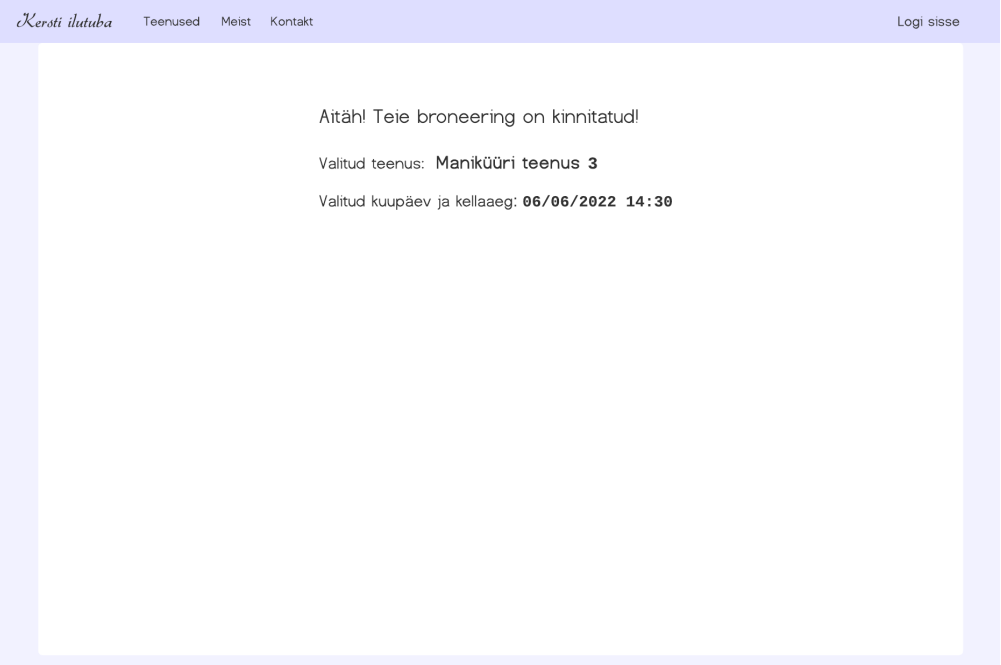

Joonis 11. Broneeringu aja valimise vaade.

Kui broneeringu kuupäev ja kellaaeg on valitud, siis kuvatakse broneeringu kliendi info vaade (Joonis 12). Selles vaates kuvatakse väljad, mille kasutaja peab enne broneeringu kinnitamist ära täitma. Väljad eesnimi, perenimi, e-mail ja telefon on kohustuslikud. Lisainfo väli ei ole kohustuslik, aga sinna võib kasutaja soovi korral lisada kommentaare. Broneeringu kinnitamiseks peab kasutaja nõustuma ka kasutustingimustega.




Joonis 12. Broneeringu kliendi info vaade.

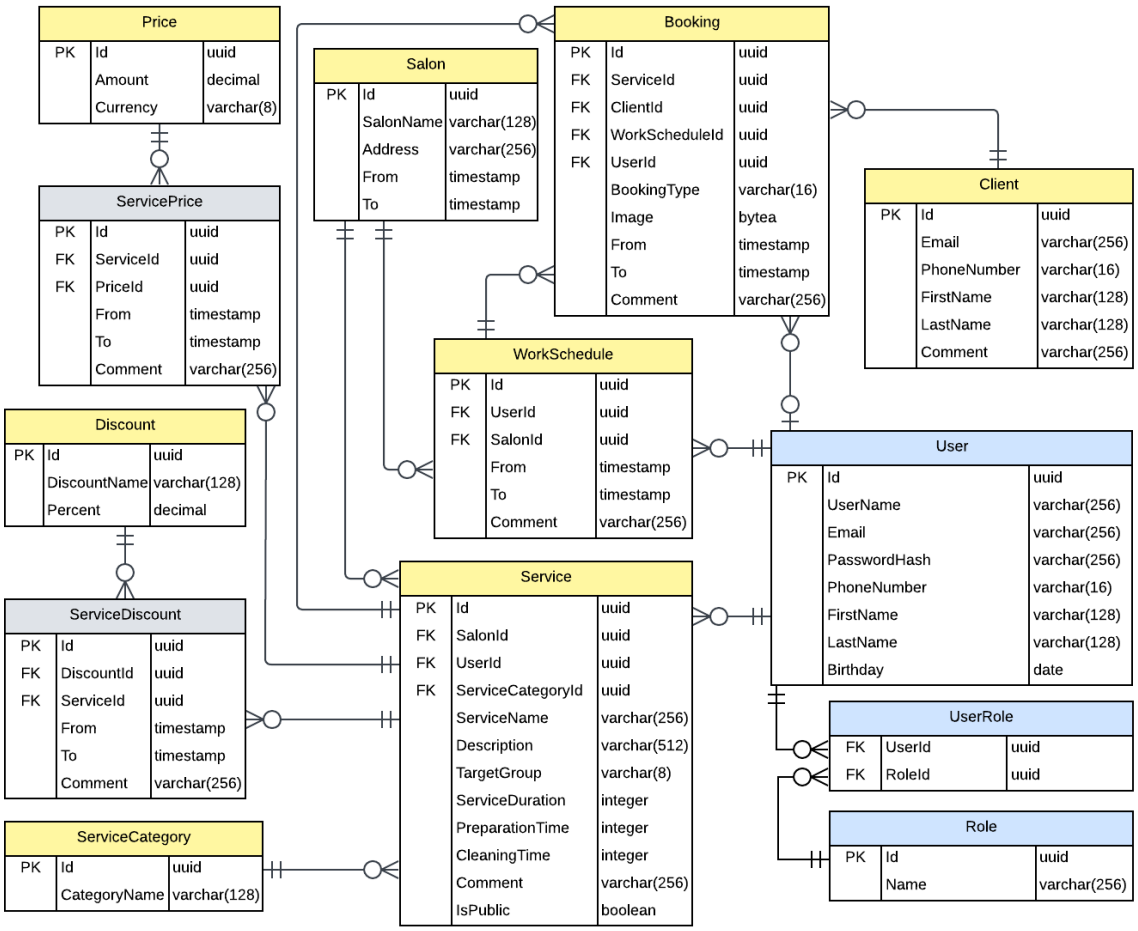
Kui kasutaja vajutab nupule „Kinnita broneering“, siis broneering kinnitatakse ning kasutajale kuvatakse broneeringu kinnitamise vaade (Joonis 13). Selles vaates teavitatakse kasutajale, et broneering on kinnitatud ja kuvatakse valitud teenus ning valitud kuupäev ja kellaaeg.




Joonis 13. Broneeringu kinnitamise vaade.

### 3.4.2 Andmebaasi disain

Lähtuvalt rakenduse nõuetest ja kasutajakogemuse disainist on koostatud andmebaasi olemi-suhte diagramm (Joonis 14). Diagrammil kujutab sinine värv kasutaja ja tema rollidega seotud olemeid, hall vahetabeleid ning kollane põhilisi olemeid.




Joonis 14. Olemi-suhte diagramm (ERD).

„User“ olem tähistab sõltuvalt rollist tavakasutajat (ehk ilusalongi klienti) või iluteenindajat. Kui „User“-i roll on iluteenindaja, siis on tal võimalik hallata oma töögraafikut („WorkSchedule“ tabel) ja teenuseid („Service“ tabel). Ta saab ka hallata teenuste kategooriaid („ServiceCategory“ tabel), teenuste hindu („Price“ tabel) ja teenuste soodustusi („Discount“ tabel). Teenuse hindade ja soodustuste kohta käivate andmete hoidmiseks on kasutusel ka vahetabelid, et talletada andmeid kõikidest teenusel olnud hindadest („ServicePrice“ tabelis) ja soodustustest („ServiceDiscount“ tabelis). Lisaks on iluteenindajal võimalik muuta oma salongi andmeid (tabel „Salon“). Iluteenindaja saab hallata ka kõiki oma teenustega seotud broneeringuid (tabel „Booking“). Kui „User“-i roll on tavaline kasutaja, siis on tal võimalik ainult broneeringuid luua, muuta ja tühistada. Tavakasutajana broneeringut luues salvestatakse broneeringusse „User“-i ID, aga kui klient loob broneeringu veebirakendusse sisse logimata, siis seda broneeringut „User“-iga siduda ei saa. „Client“ tabelis hoitakse klientide andmeid olenemata sellest kas neil on kasutaja või mitte.

Olemitel on ka atribuudid CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, aga neid pole parema loetavuse huvides olemi-suhte diagrammile lisatud. Lisaks on need andmed kasutusel peamiselt ainult andmete jälgimise ja veaotsingu eesmärgil ning ei ole otseselt seotud rakenduse äriloogikaga.

## 3.5 Analüüsi kokkuvõte

Analüüsi käigus käsitleti veebirakenduse funktsionaalseid ja mittefunktsionaalseid nõudeid ning nende põhjal loodi veebirakenduse plokkskeem (Lisa 2).

Analüüsiti erinevaid tehnoloogiaid arvestades tehnoloogiate populaarsust, autori kogemust ja tehnoloogia sobivust lõputöös käsiteldava probleemi lahendamiseks. Andmebaasisüsteemidest osutus valituks relatsiooniline andmebaasisüsteem PostgreSQL. Veebiteenuse raamistikuks valiti ASP.NET Core ning programmeerimiskeeleks C#. Klientrakenduse arendamiseks otsustati kasutada Vue.js raamistikku. Versioonihalduskeskkonnaks valiti GitHub keskkond.

Analüüsi käigus selgitati ka veebirakenduse arhitektuuri ja kasutajaliidese disaini. Lisaks kirjeldati andmebaasi olemi-suhte diagrammi, mis koostati lähtudes veebirakenduse nõuetest ja kasutajakogemuse disainist.

# 4 Veebirakenduse arendus

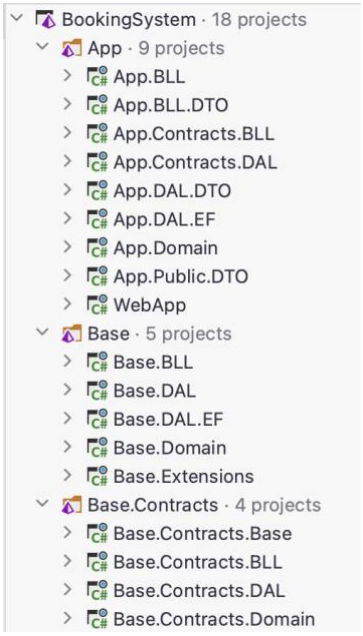
Veebirakenduse arenduse protsessi erinevad osad on jaotatud kolme suuremasse peatükki: veebiteenuse arendus, klientrakenduse arendus ja testimine.

## 4.1 Veebiteenuse arendus

Veebiteenuses ehk tagarakenduses on veebirakenduse äriloogika ning suhtlus andmebaasiga. Veebiteenuse arendamiseks on kasutatud ASP.NET Core raamistikku ja C# programmeerimiskeelt. Arendamisel on kasutatud .NET versiooni 7.0. Järgevates peatükkides on kirjeldatud erinevaid aspekte veebiteenuse arendusel.

### 4.1.1 Tagarakenduse struktuur

Tagarakenduse struktuur (Joonis 15) on koostatud veebirakenduse arhitektuuri järgi, kus veebiteenus on jagatud nelja erinevasse kihti: kontrollerid, äriloogikakiht, andmete juurdepääsu kiht ning domeenikiht.




Joonis 15. Tagarakenduse struktuur.

Veebiteenuse arhitektuurikihid on jaotatud kolme osasse: Base, Base.Contracts ja App. Base sisaldab erinevaid baasklasse koodikorduse vähendamiseks. Näiteks sisaldab see

baasrepositooriumit, mida kõik repositooriumid kasutavad. Base.Contracts sisaldab erinevaid baasliideseid, näiteks repositooriumi liidest, mis määrab milline peab repositoorium olema. App sisaldab erinevaid veebiteenuse projekte, mis sisaldavad näiteks äriloogika teenuseid, repositooriumeid ja DTO-sid.

Domeenikihis on klassid, mis väljendavad domeeniobjekte (projektis App.Domain). Andmebaasi migratsiooni tegemisel moodustatakse nendest andmebaasi olemid ning luuakse ka olemite vahelised suhted.

Andmete juurdepääsukihis on repositooriumid ja andmebaasi migratsioonid (projektis App.DAL.EF) ning repositooriumite liidesed (projektis App.Contracts.DAL). Lisaks on selles kihis ka andmeedastusobjektid (projektis App.DAL.DTO).

Äriloogikakihis on samuti andmeedastusobjektid (projektis App.BLL.DTO). Lisaks on selles ka teenused (projektis App.BLL) ja teenuste liidesed (projektis App.Contracts.BLL).

API kontrollerid asuvad projektis WebApp ning need kasutavad API päringutega tegelemiseks enda andmeedastusobjekte (projektis App.Public.DTO).

### 4.1.2 Andmebaas ja sellega ühendamine

Veebirakenduse PostgreSQL andmebaasisüsteemi ülesseadmisel arenduse ajaks on kasutatud Dockerit. Selle jaoks loodi docker-compose.yml fail (Joonis 16), kus kirjeldati andmebaasi konteinerit.

```yaml
version: "3.9"

services:
  bookingsystem-postgres:
    container_name: bookingsystem-postgres
    image: "postgres:15.2-bullseye"
    restart: unless-stopped
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    logging:
      options:
        max-size: 10m
        max-file: "3"
    ports:
      - "5432:5432"
    volumes:
      - bookingsystem-postgres-volume:/var/lib/postgresql/data

volumes:
  bookingsystem-postgres-volume:
```

Joonis 16. docker-compose.yml fail.

Pärast andmebaasi konteineri loomist ja käivitamist ühendati veebiteenus andmebaasiga. Selle jaoks lisati appsettings.json konfiguratsioonifaili andmebaasi ühendussõne (*connection string*), mis sisaldab endas andmebaasiga ühendamiseks vajalikku infot. Seda ühendussõne kasutati rakendusele andmebaasi konteksti lisamisel Program.cs failis. Seejärel tagarakenduse käivitamisel loodigi ühendus andmebaasiga.

### 4.1.3 REST API

Veebiteenus ja klientrakendus kasutavad omavaheliseks suhtluseks REST API-t , mis on API, mis vastab REST (*Representational State Transfer*) disainiprintsiipidele [39]. Veebiteenus võtab vastu klientrakenduse API päringuid ja annab kliendile tagasi API päringute vastuseid. Veebiteenuse API lõpp-punktide (*endpoint*) disainimisel lähtuti rakenduse nõuetest ja kasutajakogemuse disainist. Rakenduse nõuete täitmiseks vajalikud veebiteenuse API lõpp-punktid on välja toodud lisas 3.

API lõpp-punktid on jaotatud API kontrolleritesse. Iga API lõpp-punkti eesmärgiks on mõni operatsioon ehk meetod – GET, POST, PUT, DELETE. Kasutades meetodeid POST ja PUT antakse päringuga kaasa ka andmed JSON objektina (*API request body*). Igal API lõpp-punktil on ka oma *path* ehk osa veebiaadressist, millel võib olla lisainfot, näiteks id.

### 4.1.4 API päringute turvalisus

Eelnevalt väljatoodud API lõpp-punktide ligipääs sõltub veebirakenduse kasutaja olemasolust ja tema rollist. Sisselogimata kasutaja saab päringuid teha ainult autentimise ning broneeringu tegemisega seotud API lõpp-punktidele. Sisselogitud tavakasutaja rollis olev veebirakenduse kasutaja saab päringuid teha ka oma andmete muutmisega või broneeringutega seotud API lõpp-punktidele. Sisselogitud iluteenindaja rollis olev veebirakenduse kasutaja saab lisaks eelnevale teha päringuid kõigile ülejäänud API lõpp-punktidele, näiteks iluteenuste loomise ja muutmisega seotud API lõpp-punktidele.

Selleks, et veebiteenuses kindlaks teha kasutaja olemasolu ning tema roll, on sisselogitud kasutaja API päringutega vaja kaasa anda JWT turvamärk (*token*), mis pannakse päringu päisesse. JWT saamiseks peab kasutaja end sisse logima või kasutajaks registreerima. Sisselogimise ja registreerimise API lõpp-punktide päringute vastustes on JWT turvamärk, mida kasutaja saab edaspidi oma päringute tegemisel kasutada. Kui kasutaja poolt tehtud päringu päises olev JWT pole korrektne, ta pole sisse loginud või tal puudub

vajalik roll mõne päringu tegemiseks, siis kasutaja andmetele ligi ei pääse ning teda teavitatakse sellest päringu vastuses.

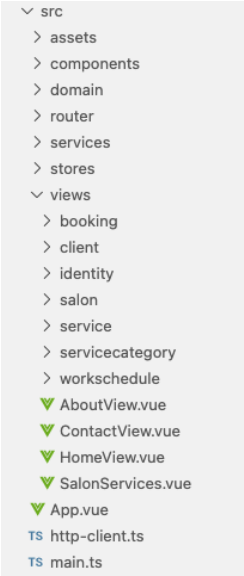
## 4.2 Klientrakenduse arendus

Klientrakenduse ehk esirakenduse arendamiseks on kasutatud Vue.js raamistikku koos TypeScript programmeerimiskeelega. Arendamisel on kasutatud Vue.js versiooni 3.2.47. Järgnevates peatükkides on kirjeldatud erinevaid aspekte klientrakenduse arendusel.

### 4.2.1 Klientrakenduse struktuur

Klientrakenduse struktureerimiseks on rakenduse erinevad osad jagatud järgnevatesse kaustadesse (Joonis 17):

- Assets – selles kaustas on kõik klientrakendusega seotud pildid, näiteks ilusalongi logo.
- Components – erinevad klientrakenduse väikesed osad ehk komponendid, näiteks veebilehe päis.
- Domain – sisaldab API päringutes kasutatavate objektide liideseid.
- Router – sisaldab faili index.ts, milles luuakse klientrakenduse ruuter, mida kasutatakse vaadete vahel navigeerimiseks.
- Services – selles kaustas on erinevad teenused, mis sisaldavad tagarakenduse API lõpp-punktidele päringuid tegevaid meetodeid.
- Stores – klientrakenduse andmehoidlad, kuhu saab salvestada andmeid, näiteks saab andmehoidlasse salvestada kasutaja JWT turvamärgi.
- Views – kõik klientrakenduse vaated, millest suurem osa on jagatud eraldi kaustadesse.




Joonis 17. Klientrakenduse struktuur.

### 4.2.2 Suhtlus veebiteenusega

Klientrakendus suhtleb veebiteenusega API päringute abil. Selle jaoks on klientrakenduses loodud http-client.ts fail (Joonis 18), kus on kastutatud axios teeki, mis lihtsustab API päringute tegemist. Axios instantsi loomisel määratakse veebiteenuse baasURL ja päis, mis on igal API päringul kaasas.

```
1   import axios from "axios";
2
3   export const httpClient = axios.create({
4       baseURL: "https://localhost:7164/api/v1",
5       headers: {
6           "Content-type": "application/json"
7       }
8   });
9
10  export default httpClient;
```

Joonis 18. http-client.ts fail.

Failis http-client.ts olev httpClient on kasutusel kõikides klientrakenduse teenustes asuvates meetodites, mis teevad päringuid veebiteenuse API lõpp-punktidele.

### 4.2.3 Turvalisus klientrakenduses

Rakendusse sisse logides või kasutajaks registreerides saab kasutaja veebiteenuselt JWT turvamärgi. Turvamärk salvestatakse klientrakenduse andmehoidlasse identity.ts ning enne igat API päringu tegemist lisatakse andmehoidlas olev turvamärk API päringu

päisesse. Kui kasutaja logib välja või tema turvamärk aegub, siis see kustutakse ära ka identity.ts andmehoidlast.

Klientrakenduses on sisselogimata kasutajal võimalik näha vaid broneeringu tegemise ja autentimisega seotud vaateid. Sisselogitud tavakasutaja rollis olev kasutaja saab näha ka oma broneeringutega ja andmetega seotud vaateid. Sisselogitud iluteenindajal on võimalik näha lisaks eelnevalt mainitud vaadetele ka kõiki tema ilusalongi ja tööga seotud vaateid.

### 4.2.4 Broneeringu loomise vaated

Lähtuvalt veebirakenduse analüüsis kirjeldatud kasutajakogemuse disainist on loodud klientrakenduse broneeringu loomise vaated. Kasutajakogemuse disaini järgi tehti vaadete komponendid (näiteks loodi vajalikud kliendi info väljad), kuid parema kasutajakogemuse huvides muudeti veidi vaadete kujundust. Loodud broneeringu tegemise vaated on lisas 4.

## 4.3 Testimine

Käesoleva lõputöö kontekstis mõeldakse testimise all veebirakenduse manuaalset testimist.

Testimise käigus testiti veebirakenduse nõuetele vastavust. Selle käigus veenduti, et ilusalongi broneeringu tegemine töötab korrektselt ning rakendusse sisselogitud kasutajatel on võimalik oma broneeringuid hallata. Testiti ka kõiki vaid iluteenindajale kättesaadavaid funktsionaalsusi ehk töögraafiku ja iluteenuste haldamist ning salongi info kättesaadavust ja muutmist. Lisaks testiti ka seda, et sisselogitud kasutajatel on võimalik ligi saada vaid endaga seotud andmetele ning ka seda, et iluteenindaja salongi ja tööga seotud andmetele pääseb ligi ainult iluteenindaja rollis olev kasutaja.

Lõputöö autoril on plaanis luua veebirakenduse testimiseks ka automaattestid, kuid paraku see käesoleva lõputöö skoopi ei mahtunud.

# 5 Hinnang loodud veebirakendusele

Loodud veebirakendust saab hinnata veebirakenduse nõutele vastavuse järgi. Lisaks võib hinnata tulevaste edasiarenduste võimalikkust.

## 5.1 Valminud veebirakenduse nõutele vastavus

Veebirakenduse arenduse käigus said täidetud kõik mittefunktsionaalsed nõuded ja suurem osa funktsionaalsetest nõuetest.

Loodud veebirakendus on eestikeelne ning seda on võimalik lihtsasti kasutada nii arvutis kui ka nutiseadmes. Veebirakendus võimaldab mugavalt teha iluteenuste broneeringuid. Sisselogitud kasutajatel on võimalik oma broneeringuid ka näha ja muuta. Iluteenindaja saab veebirakenduses hallata oma iluteenuste, töögraafiku ja salongiga seotud andmeid.

Funktsionaalsetest nõuetest jäid saavutamata kasutaja andmete muutmise nõue, tehtud töödest piltide lisamise ja nägemise nõuded ning kliendi ajaloo nägemise ja kliendi kohta kommentaaride lisamise nõuded.

## 5.2 Edasiarenduse võimalused

Veebirakendust edasi arendades on kõige olulisem esmalt valmis saada täitmata jäänud funktsionaalsed nõuded. Lisaks on plaanis teha rakendusele ka automaattestid.

Rakenduse võimalikeks edasiarendusteks on uute funktsionaalsuste lisamine, näiteks oleks võimalik rakendusse lisada broneeringute e-maili teavitused, kinkekaartide soetamise võimalus, arvete koostamine ja saatmine ning iluteenindajale tema salongi ja iluteenustega seotud statistika kuvamine.

Tänu veebiteenuse ja klientrakenduse eraldatusele on üheks võimalikuks edasiarenduseks ka uue klientrakenduse ehk ilusalongi aegade broneerimist võimaldava nutirakenduse loomine.

# 6 Kokkuvõte

Käesoleva bakalaureusetööga loodi veebirakendus ilusalongi teenuste aegade broneerimise protsessi lihtsustamiseks. Töö käigus kirjeldati lahendatavat probleemi ning pandi paika loodava lahenduse skoop. Veebirakenduse analüüsis toodi välja rakenduse nõuded, valiti probleemi lahendamiseks sobivad tehnoloogiad, analüüsiti veebirakenduse arhitektuuri ning kasutajakogemuse ja andmebaasi disaini. Analüüsile järgnes veebirakenduse arendus, mille käigus valmisid nii veebiteenus kui ka klientrakendus, täideti suurem osa rakenduse nõuetest ning testiti loodud veebirakendust.

Lõputöös püstitatud eesmärk täideti ning valminud veebirakendus ilusalongi aegade broneerimiseks lahendab lõputöös kirjeldatud probleemi. Tänu loodud veebirakendusele ei ole iluteenuste broneeringute tegemine enam aeganõudev protsess, kuna nüüd saavad kliendid iseseisvalt igal ajahetkel kiirelt ja mugavalt broneeringuid teha. Lisaks lihtsustab valminud veebirakendus iluteenindaja tööd ning tänu sellele on tal võimalik panustada rohkem aega iluteenuste teostamisele.

# Kasutatud kirjandus


[1] „Saloninfra" [Võrgumaterjal]. Loetud aadressil: https://www.saloninfra.ee/saloninfra [Kasutatud 26.03.23]

[2] „Booklux" [Võrgumaterjal]. Loetud aadressil: https://www.booklux.com/et [Kasutatud 26.03.23]

[3] „SalonLife" [Võrgumaterjal]. Loetud aadressil: https://www.salon.life/ [Kasutatud 27.03.23]

[4] "Mobile Apps vs Web Apps: Which is the Better Option?" [Võrgumaterjal]. Loetud aadressil: https://sagaratechnology.medium.com/mobile-apps-vs-web-apps-which-is-the-better-option-868106c88730 [Kasutatud 28.03.23]

[5] "Database Selection & Design (Part I) " [Võrgumaterjal]. Loetud aadressil: https://f5sal.medium.com/database-selection-design-part-i-346c74291d3d [Kasutatud 10.03.23]

[6] "What's the Difference? Relational vs Non-Relational Databases" [Võrgumaterjal]. Loetud aadressil: https://insightsoftware.com/blog/whats-the-difference-relational-vs-non-relational-databases/ [Kasutatud 10.03.23]

[7] "Relational vs. Non-Relational Databases" [Võrgumaterjal]. Loetud aadressil: https://www.mongodb.com/compare/relational-vs-non-relational-databases [Kasutatud 10.03.23]

[8] "2022 Developer Survey. Databases" [Võrgumaterjal]. Loetud aadressil: https://survey.stackoverflow.co/2022/#most-popular-technologies-database-prof [Kasutatud 10.03.23]

[9] "Which Modern Database Is Right For Your Use Case? " [Võrgumaterjal]. Loetud aadressil: https://www.integrate.io/blog/which-database/ [Kasutatud 10.03.23]

[10] "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems" [Võrgumaterjal]. Loetud aadressil: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems [Kasutatud 10.03.23]

[11] "Microsoft SQL Server Pros and Cons" [Võrgumaterjal]. Loetud aadressil: https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/ [Kasutatud 12.03.23]

[12] "The Most Popular Backend Frameworks for Web Development" [Võrgumaterjal]. Loetud aadressil: https://radixweb.com/blog/best-backend-frameworks [Kasutatud 13.03.23]

[13] "Top 5 backend frameworks" [Võrgumaterjal]. Loetud aadressil: https://ryax.tech/top5-backend-frameworks/ [Kasutatud 13.03.23]

[14] "Overview of ASP.NET Core" [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0 [Kasutatud 13.03.23]

[15] "Top 10 Advantages of .NET Core" [Võrgumaterjal]. Loetud aadressil: https://www.fortech.ro/top-advantages-net-core/ [Kasutatud 13.03.23]

[16] “Language Integrated Query (LINQ) (C#)“ [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ [Kasutatud 14.03.23]

[17] “Django“ [Võrgumaterjal]. Loetud aadressil: https://www.djangoproject.com/ [Kasutatud 14.03.23]

[18] “What is Python? Executive Summary“ [Võrgumaterjal]. Loetud aadressil: https://www.python.org/doc/essays/blurb/ [Kasutatud 14.03.23]

[19] “Spring Boot“ [Võrgumaterjal]. Loetud aadressil: https://spring.io/projects/spring-boot [Kasutatud 14.03.23]

[20] “Spring Boot Security Auto-Configuration“ [Võrgumaterjal]. Loetud aadressil: https://www.baeldung.com/spring-boot-security-autoconfiguration [Kasutatud 14.03.23]

[21] “Spring Security“ [Võrgumaterjal]. Loetud aadressil: https://docs.spring.io/spring-security/reference/index.html [Kasutatud 14.03.23]

[22] “The Good and the Bad of Java Programming“ [Võrgumaterjal]. Loetud aadressil: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/ [Kasutatud 14.03.23]

[23] “Top 15 Programming Languages with the Largest Developer Communities“ [Võrgumaterjal]. Loetud aadressil: https://www.dice.com/career-advice/top-15-programming-languages-with-the-largest-developer-communities [Kasutatud 04.04.23]

[24] “List of 10 Best Front end Frameworks to Use For Web Development“ [Võrgumaterjal]. Loetud aadressil: https://www.monocubed.com/blog/best-front-end-frameworks/ [Kasutatud 14.03.23]

[25] “Aurelia“ [Võrgumaterjal]. Loetud aadressil: https://aurelia.io/ [Kasutatud 14.03.23]

[26] “Technical Benefits“ [Võrgumaterjal]. Loetud aadressil: https://aurelia.io/docs/overview/technical-benefits/ [Kasutatud 14.03.23]

[27] “2022 Developer Survey. Web frameworks and technologies“ [Võrgumaterjal]. Loetud aadressil: https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe [Kasutatud 14.03.23]

[28] “Vue“ [Võrgumaterjal]. Loetud aadressil: https://vuejs.org/ [Kasutatud 14.03.23]

[29] “Top 10 Practical Benefits of Vue.js for Web Development“ [Võrgumaterjal]. Loetud aadressil: https://www.techmagic.co/blog/benefits-of-vuejs/ [Kasutatud 14.03.23]

[30] “Introducing JSX“ [Võrgumaterjal]. Loetud aadressil: https://reactjs.org/docs/introducing-jsx.html [Kasutatud 15.03.23]

[31] “Best Version Control Hosting Software“ [Võrgumaterjal]. Loetud aadressil: https://www.g2.com/categories/version-control-hosting [Kasutatud 17.03.23]

[32] “The Moz Top 500 Websites“ [Võrgumaterjal]. Loetud aadressil: https://moz.com/top500 [Kasutatud 17.03.23]

[33] “GitHub“ [Võrgumaterjal]. Loetud aadressil: https://www.crunchbase.com/organization/github [Kasutatud 17.03.23]

[34] “GitHub“ [Võrgumaterjal]. Loetud aadressil: https://github.com/ [Kasutatud 17.03.23]

[35] “GitLab“ [Võrgumaterjal]. Loetud aadressil: https://about.gitlab.com/ [Kasutatud 17.03.23]

[36] “Bitbucket“ [Võrgumaterjal]. Loetud aadressil: https://www.crunchbase.com/organization/bitbucket [Kasutatud 17.03.23]

[37] “Compare plans and pricing“ [Võrgumaterjal]. Loetud aadressil: https://www.atlassian.com/software/bitbucket/pricing [Kasutatud 17.03.23]

[38] Robert C. Martin, “Clean Architecture”, 2017

[39] “What is a REST API? “ [Võrgumaterjal]. Loetud aadressil: https://www.ibm.com/topics/rest-apis [Kasutatud 12.04.23]

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

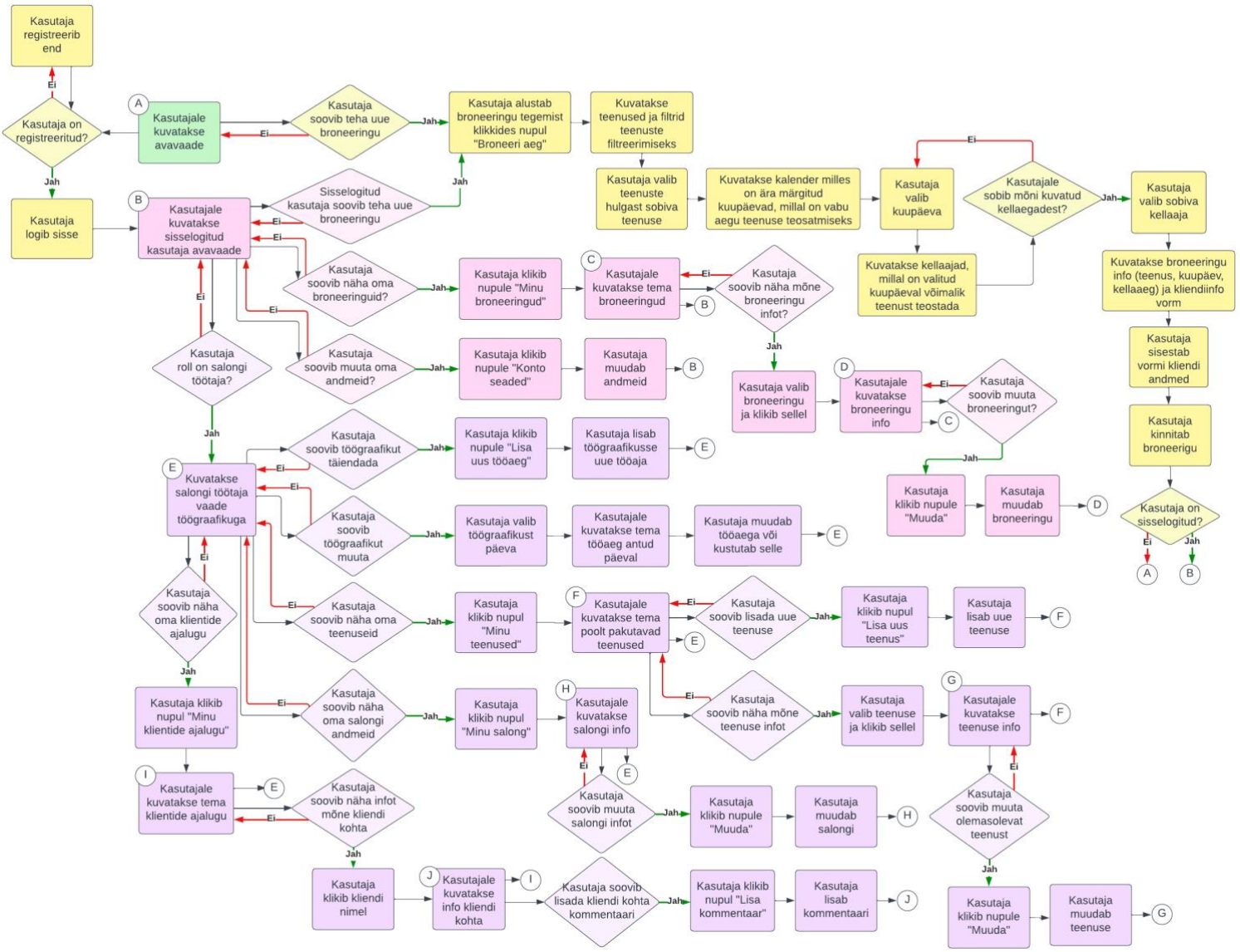
Mina, Diana Christine Tarro

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Ilusalongi aegade broneerimise veebirakenduse arendus“, 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.

24.04.23

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 – Plokkskeem

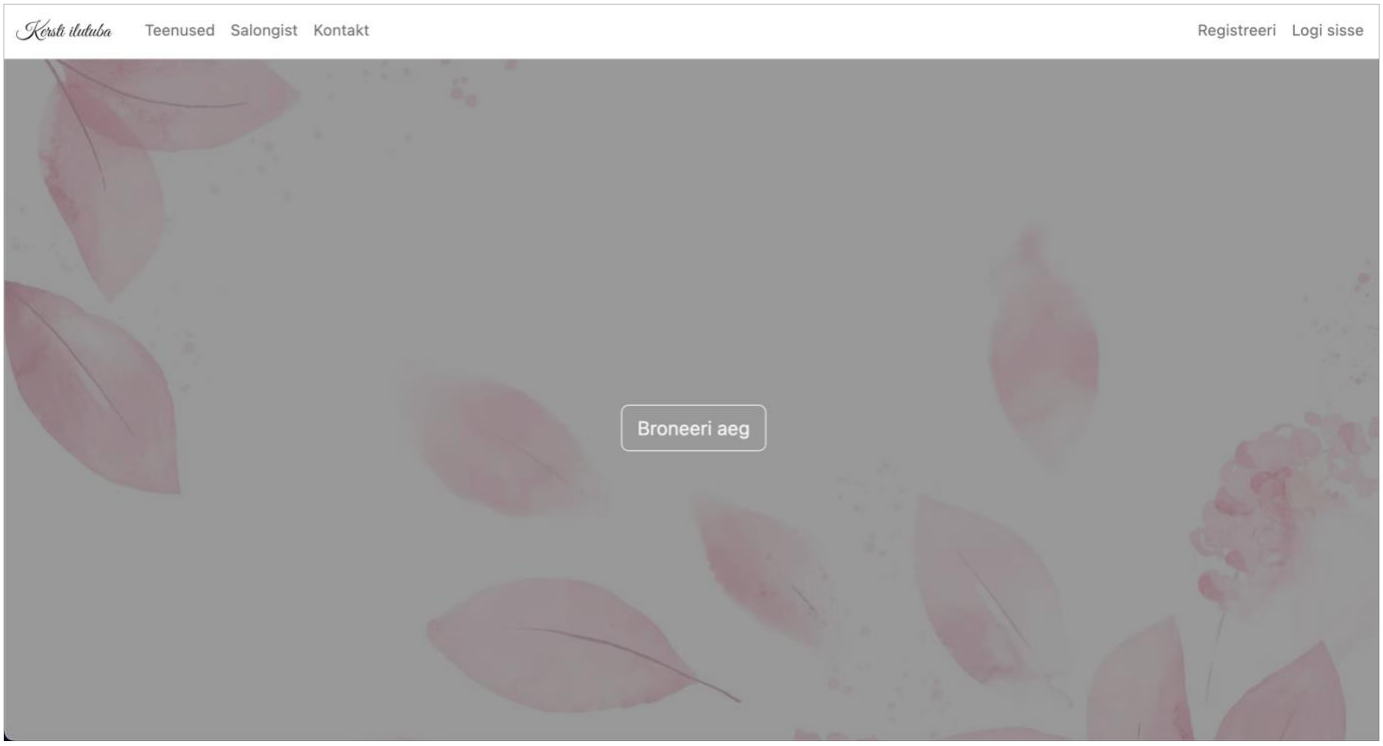


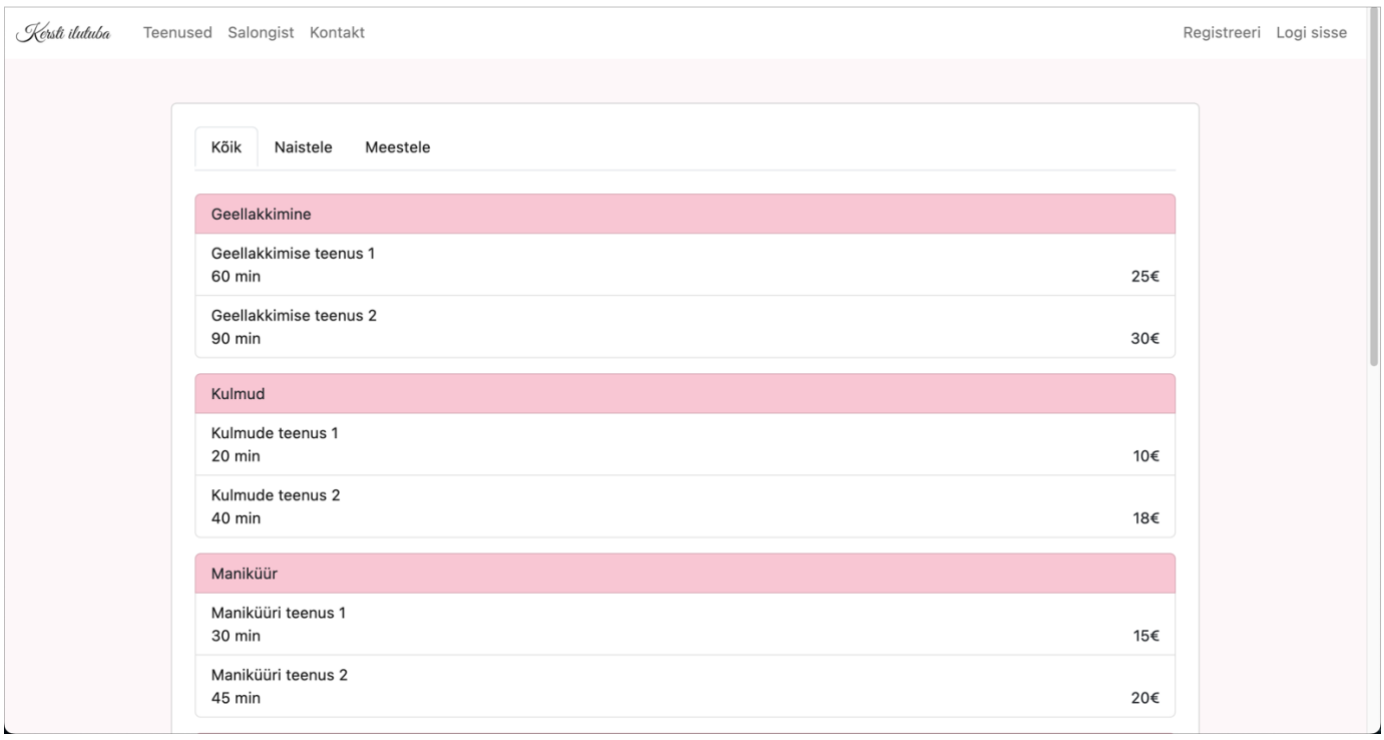
# Lisa 3 – Veebiteenuse API lõpp-punktid

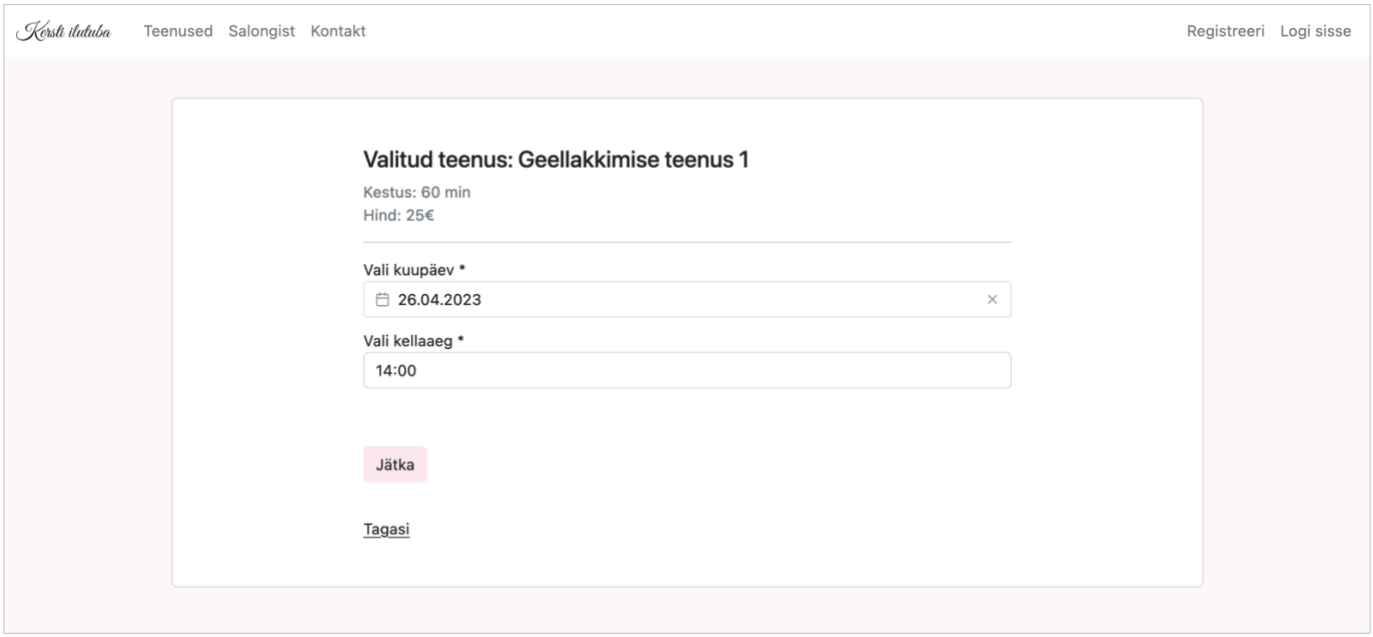
| **AccountController** | | |
|---|---|---|
| Kasutaja loomine | POST | */identity/account/register* |
| Sisse logimine | POST | */identity/account/login* |
| Välja logimine | POST | */identity/account/logout* |
| Kasutaja andmete muutmine | PUT | */identity/account/data* |
| **ServicesController** | | |
| Teenusete pärimine | GET | */services?targetgroup= &servicecategory=* |
| Teenuse pärimine | GET | */services/{id}* |
| Teenuse lisamine | POST | */services* |
| Teenuse muutmine | PUT | */services/{id}* |
| Teenuse kustutamine | DELETE | */services/{id}* |
| **ServiceCategoriesController** | | |
| Teenuse kategooriate pärimine | GET | */servicecategories* |
| Teenuse kategooria lisamine | POST | */servicecategories* |
| Teenuse kategooria muutmine | PUT | */servicecategories/{id}* |
| Teenuse kategooria kustutamine | DELETE | */servicecategories/{id}* |
| **BookingsController** | | |
| Broneeringute pärimine | GET | */bookings?datesfrom= &datesto=* |
| Broneeringu pärimine | GET | */bookings/{id}* |
| Broneeringu lisamine | POST | */bookings* |
| Broneeringu muutmine | PUT | */bookings/{id}* |
| Broneeringu kustutamine | DELETE | */bookings/{id}* |
| **WorkScheduleController** | | |
| Töögraafikus antud teenuse jaoks vabade aegade pärimine | GET | */workschedules/availabletimes/{year}/{month}/{serviceid}* |
| Töögraafiku tööaegade pärimine | GET | */workschedules?datesfrom= &datesto=* |
| Töögraafiku tööaja pärimine | GET | */workschedules/{id}* |
| Töögraafikusse tööaja lisamine | POST | */workschedules* |
| Töögraafikus tööaja muutmine | PUT | */workschedules/{id}* |
| Töögraafikus tööaja kustutamine | DELETE | */workschedules/{id}* |
| **SalonsController** | | |
| Salongi info pärimine | GET | */salon* |

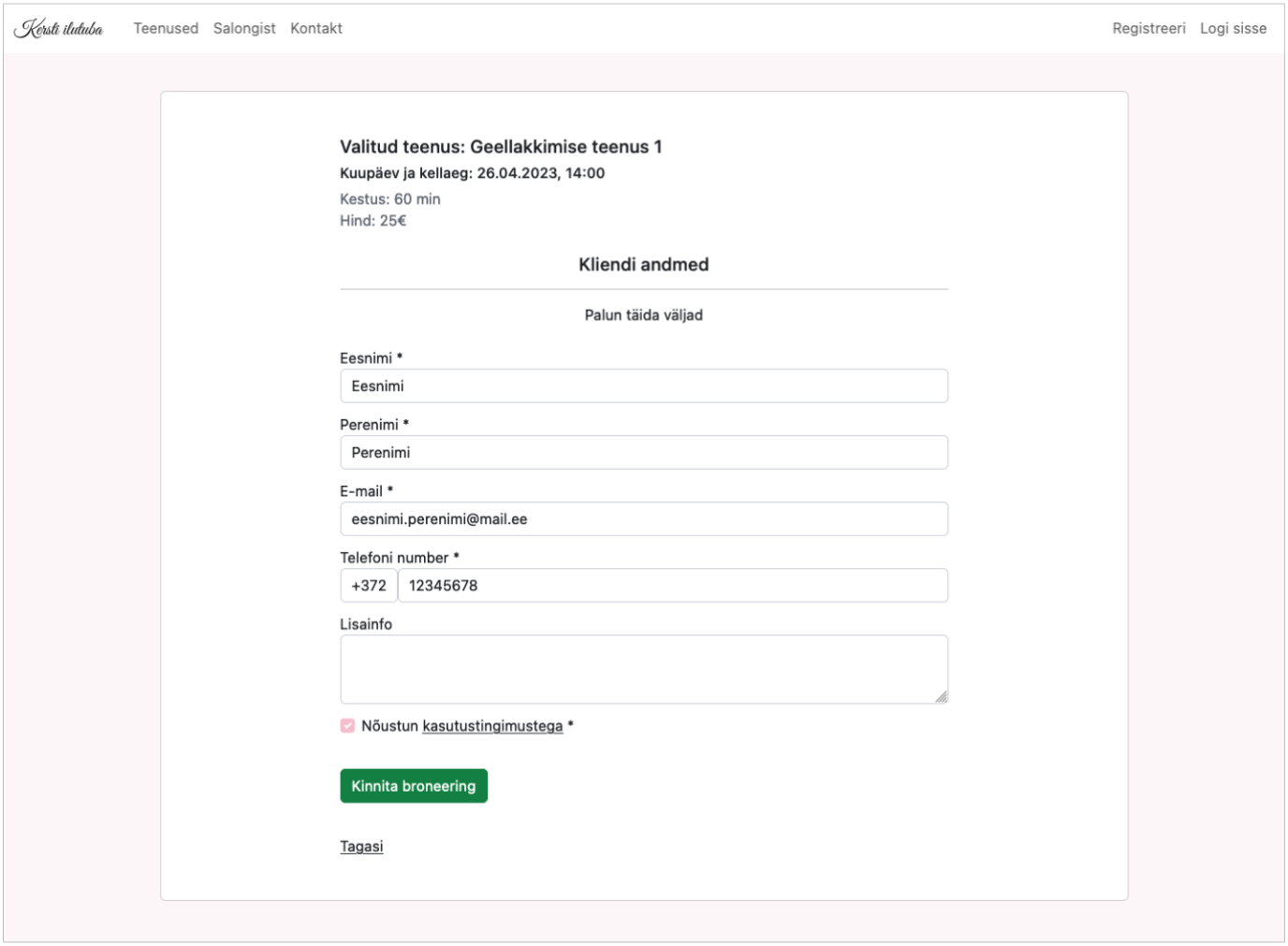
| | | |
|---|---|---|
| Salongi info muutmine | PUT | */salon* |
| **ClientsController** | | |
| Klientide pärimine | GET | */clients* |
| Kliendi pärimine | GET | */clients/{id}* |
| Kliendi loomine | POST | */clients* |
| Kliendi muutmine | PUT | */clients/{id}* |

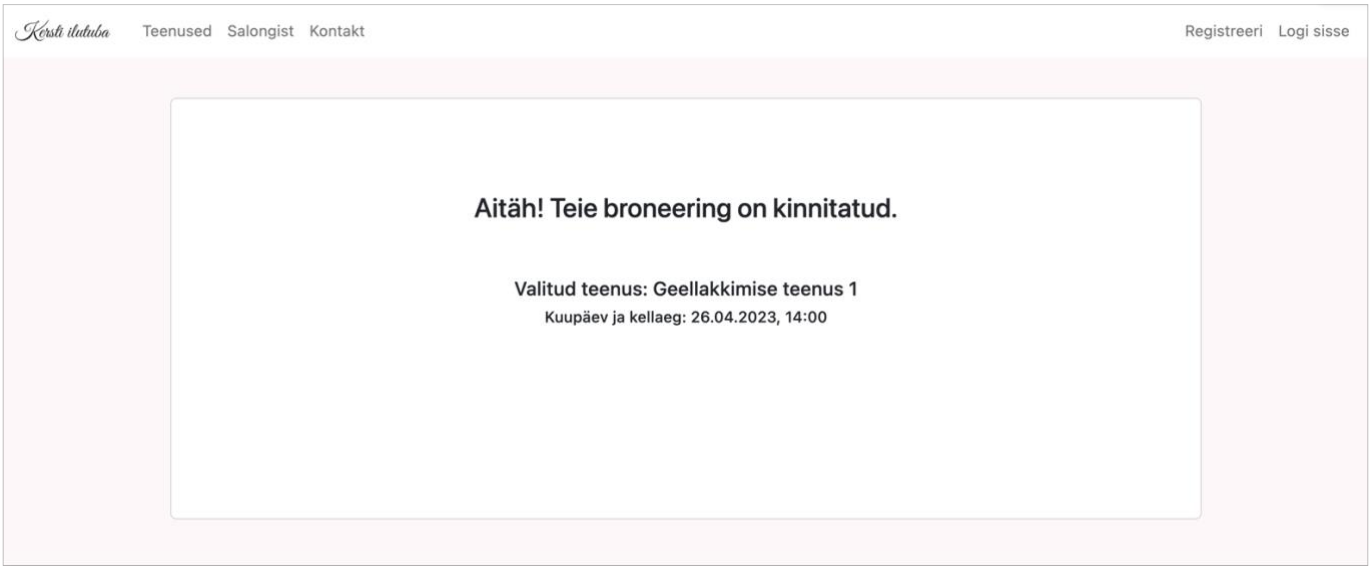
# Lisa 4 – Broneeringu loomise vaated




Kersti iluduba
Teenused
Salongist
Kontakt
Registreeri
Logi sisse
Valitud teenus: Geellakkimise teenus 1
Kestus: 60 min
Hind: 25€
Vali kuupäev *
26.04.2023
Vali kellaaeg *
14:00
Jätka
Tagasi


Kersti iluduba
Teenused
Salongist
Kontakt
Registreeri
Logi sisse
Valitud teenus: Geellakkimise teenus 1
Kuupäev ja kellaaeg: 26.04.2023, 14:00
Kestus: 60 min
Hind: 25€
Kliendi andmed
Palun täida väljad
Eesnimi *
Eesnimi
Perenimi *
Perenimi
E-mail *
eesnimi.perenimi@mail.ee
Telefoni number *
+372
12345678
Lisainfo
Nõustun kasutustingimustega *
Kinnita broneering
Tagasi


Kersti iluduba
Teenused
Salongist
Kontakt
Registreeri
Logi sisse
Aitäh! Teie broneering on kinnitatud.
Valitud teenus: Geellakkimise teenus 1
Kuupäev ja kellaeg: 26.04.2023, 14:00