TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Oskar Laak 206366IADB

Kohila Meedik OÜ broneerimissüsteemi prototüüp

Bakalaureusetöö

Juhendaja: Kersti Antoi
Magistrikraad

Tallinn 2025

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: Oskar Laak

06.01.2025

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua Kohila Meedik OÜ hambaravikeskusele broneeringusüsteemi prototüüp veebirakenduse näol. Broneeringusüsteem võimaldab klientidel teha iseseisvalt broneeringuid, pakkudes seeläbi ka täiendavat mugavust hambaravikeskuse registratuurile.

Töö teoreetilises osas kirjeldatakse lahendatavat probleemi, analüüsitakse selle aktuaalsust ja kirjeldatakse hetkeolukorda. Käsitletakse võimalikke lahendusi ning pannakse nende alusel paika loodava lahenduse skoop ning esitatakse rakendusele nõuded. Analüüsitakse samme rakenduse realiseerimiseks ning põhjendatakse ära rakendatavate tehnoloogiate valik. Kirjeldatakse rakenduse arhitektuuri ning kasutajaliidese toimingut.

Töö praktilises osas luuakse broneeringusüsteemi prototüüp. Prototüüp sisaldab nii esi- kui ka tagarakendust. Arenduse faasis analüüsitakse ja põhjendatakse erinevaid tehnilisi valikuid. Selgitatakse nii tagarakenduse ülesehitust kui ka esirakenduse loodud vaateid. Valminud lahenduse puhul hinnatakse selle nõuetele vastavust ning tuuakse välja rakenduse peamised nõrkused. Kirjeldatakse edasiarenduse võimalusi ning hinnangulisi rakenduse kasutuselevõtu majanduslikke kulusid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 6 peatükki, 21 joonist, 1 tabel.

Abstract

Kohila Meedik OÜ Booking System Prototype

The aim of this thesis is to create a booking system prototype in the form of a web application for Kohila Meedik OÜ dental clinic. The booking system allows clients to independently book their dental appointments, while also providing additional convenience for the dental clinic's reception staff.

The theoretical part of the thesis describes the problem to be solved, analyzes its relevance, and outlines the current situation. It examines possible solutions, defines the scope of the proposed system and presents the application requirements. The steps necessary for implementing the application are analyzed, and the choice of technologies is justified. The theoretical part concludes with an overview of the application's architecture and UI (*User Interface*) flow.

In the practical part of the thesis, the booking system prototype is developed. The prototype consists of both front-end and back-end application. During the development phase, various technical choices are analyzed and justified. Back-end structure and front-end UI decisions are explained with reasoning. The final solution is evaluated for compliance with the requirements, while also identifying its main weaknesses. Additionally, possibilities for further development are presented, along with an estimation of the economic costs of deploying the application.

The thesis is in Estonian and contains 38 pages of text, 6 chapters, 21 figures, 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – rakendusliides
BLL	*Business Logic Layer* – äriloogikakiht
CSS	*Cascading Style Sheets* – kaskaadlaadistik
DI	*Dependency Injection* – sõltuvuste sisestamine
DTO	*Data Transfer Object* – andmeedastusobjekt
HTTP	*Hypertext Transfer Protocol* – hüperteksti edastusprotokoll
ISO	*International Organization for Standardization* – rahvusvaheline standardiorganisatsioon
JSON	*JavaScript Object Notation* – JavaScripti andmeedastusvorming
LTS	*Long Term Support* – pikaajaline tugi
MIT	*Massachusetts Institute of Technology* – Massachusettsi tehnoloogiainstituut
REST	*Representational State Transfer* – esitusoleku edastus
UI	*User Interface* – kasutajaliides

Sisukord

1 Sissejuhatus 10
1.1 Metoodika 10
2 Käsitletav probleem 11
2.1 Hetkeolukord 11
2.2 Probleemi aktuaalsus 11
2.3 Võimalikud lahendused 11
2.4 Loodava lahenduse skoop 13
3 Loodava rakenduse analüüs 14
3.1 Funktsionaalsed nõuded 14
3.2 Mittefunktsionaalsed nõuded 14
3.3 Sisemise süsteemiga integreerimine 15
3.3.1 Dentase broneeringuliidese sobilikkus 16
3.4 Autentimine 16
3.4.1 Autentimisteenuse valik 17
3.4.2 Smart-ID integreerimine 18
3.5 Tehnoloogia valik 18
3.5.1 Esirakenduse tehnoloogia valik 19
3.5.2 React komponentteegi valik 19
3.5.3 Tagarakenduse tehnoloogia valik 20
3.5.4 Versioonihalduskeskkonna valik 21
3.6 Veebirakenduse arhitektuur 21
3.6.1 Tagarakenduse arhitektuur 21
3.6.2 Esirakenduse arhitektuur 22
3.7 Veebirakenduse disain 23
4 Rakenduse arendus 26
4.1 Tagarakenduse arendus 26
4.1.1 Kontrollerite lõpp-punktide disain 26
4.1.2 Dentase broneeringuliidese analüüs 27
4.1.3 Dentase broneeringuliidese kasutamine 29

4.1.4 Andmete vahemälus hoiustamise analüüs 30
4.1.5 Andmete vahemälus hoiustamise implementatsioon 31
4.1.6 Autentimise analüüs 32
4.1.7 Autentimise implementatsioon 33
4.2 Esirakenduse arendus 34
4.2.1 Suhtlus tagarakendusega 34
4.2.2 Esirakenduse vaated 35
4.2.3 Esirakenduse vaated mobiilseadmetes 41
5 Hinnang loodud rakendusele 44
5.1 Nõuetele vastavus 44
5.2 Nõrkused 44
5.3 Edasiarenduse võimalused 45
5.4 Rakenduse kasutuselevõtu majanduslikud kulud 46
5.4.1 Dentase broneeringuliidese kulu 46
5.4.2 Autentimisteenuste kulu 46
5.4.3 Veebirakenduse hoiustamise kulu 47
5.4.4 Kulude kokkuvõte 47
6 Kokkuvõte 48
Kasutatud kirjandus 49
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 51
Lisa 2 – Dentase veebirakenduse vaade 52
Lisa 3 – Loodud veebirakenduse mobiilivaated 53

Jooniste loetelu

Joonis 1. Kasutaja autentimise plokkskeem. 23
Joonis 2. Broneeringusüsteemi kasutamise plokkskeem. 24
Joonis 3. Dentase broneeringuliidese päringu sisendi struktuur. 28
Joonis 4. Dentase broneeringuliidese päringu väljundi struktuur. 28
Joonis 5. Andmete vahemälus hoiustamise konfiguratsioon. 31
Joonis 6. Sessiooni andmetesse isikuandmete salvestamine. 33
Joonis 7. Sessiooni andmetest isikuandmete lugemine. 33
Joonis 8. Sessiooni andmetest isikuandmete eemaldamine. 33
Joonis 9. Esirakenduse päringute konfiguratsioon. 35
Joonis 10. Broneeringusüsteemi esimene autentimisvaade. 36
Joonis 11. Broneeringusüsteemi teine autentimisvaade. 36
Joonis 12. Broneeringusüsteemi teenuste valimise vaade. 37
Joonis 13. Broneerimissüsteemi arstide valimise vaade. 38
Joonis 14. Broneerimissüsteemi broneeritava aja valimise vaade. 38
Joonis 15. Broneerimissüsteemi isikuandmete sisestamise vaade. 39
Joonis 16. Broneerimissüsteemi broneeringu kinnitamise vaade. 40
Joonis 17. Broneeringusüsteemi eduka broneeringu sooritamise vaade. 41
Joonis 18. Broneerimissüsteemi broneeringu ülevaate vaade. 41
Joonis 19. Ekraanilaiusel põhineva stiilireegli kasutamise näide. 42
Joonis 20. WindowWidthContext.ts fail. 42
Joonis 21. WindowWidthProvider.tsx fail. 43

Tabelite loetelu

Tabel 1. Tagarakenduse lõpp-punktid. 27

1 Sissejuhatus

Broneeringute tegemine läbi veebirakenduse on tänapäeva maailmas tavapärane protsess. Broneeringusüsteemid aitavad inimestel teha iseseisvalt broneeringuid, lihtsustades seeläbi ka teenuse aegu pakkuvate ettevõtete tööd.

Kohilas asuv hambaravikeskus Kohila Meedik OÜ hetkeseisuga oma klientidele iseseisvat aegade broneerimise võimalust ei paku. Käesoleva lõputöö eesmärk on luua eelnimetatud ettevõttele broneerimissüsteemi prototüüp, mille abil oleks võimalik tulevikus klientidele seda teenust pakkuda.

1.1 Metoodika

Käesolevas lõputöös sõnastatakse esmalt käsitletav probleem. Analüüsitakse hetke olukorda ning sellest tulenevaid probleeme. Seejärel sõnastatakse eesmärk ning analüüsitakse samme selleni jõudmiseks. Analüüsi käigus arvestatakse erinevate võimalike lahendustega ning tehakse nende seast põhjendatud otsus. Loodava broneeringusüsteemi analüüsil saab olema oluline, et broneeringusüsteem lihtsustaks nii kliendi kui ka registratuuri tööd.

Loodavat lahendust analüüsitakse. Määratakse broneeringusüsteemi prototüübi funktsionaalsed ning mittefunktsionaalsed nõuded. Analüüsi käigus võrreldakse erinevaid tehnoloogiaid ning valitakse nende seast probleemi lahendamiseks parimad. Valikuid põhjendatakse. Esirakenduse puhul analüüsitakse selle ülesehitust, suhtlust tagarakendusega, kasutusmugavust ning kasutajaliidese disaini. Tagarakenduse puhul analüüsitakse selle ülesehitust, kasutajate autentimist, päringute optimeerimist ja hambaravikeskuses kasutusel oleva süsteemiga integreerimist.

Rakenduse arendusfaasis kirjeldatakse rakenduse arendusprotsessi. Veebirakendust testitakse lõputöö raames ajaliste piirangute tõttu vaid käsitsi.

Lõputöö käigus valminud rakendusele antakse hinnang ning tuuakse välja peamised nõrgad kohad ning edasiarenduse võimalused.

2 Käsitletav probleem

Järgnevalt kirjeldatakse lõputöö käigus käsitletavat probleemi. Tuuakse välja hetkeolukord, probleemi aktuaalsus, võimalikud lahendused ja loodava lahenduse skoop.

2.1 Hetkeolukord

Kohila Meedik OÜ hambaravikeskus võimaldab klientidele aegade broneerimisi kahel viisil. Aeg on võimalik broneerida kas kohapeal või telefoni teel. Mõlemal juhul peab aga klient arvestama kliiniku lahtiolekuaegadega ning olema broneeringu tegemisel otseses kontaktis registratuuriga. Teisisõnu puudub kliendil võimekus teha broneeringuid iseseisvalt, talle sobival ajal. Lisaks raskendab hetke olukord registratuuri tööd, kuna aegade kokkuleppimine on tülikas ning kasutab ära palju inimese tööaega, mida saaks rakendada muudele tegevustele efektiivsemalt. Registratuur on huvitatud panustama võimalikult vähe aega tegevustele, mis on kliendil võimalik teha ära iseseisvalt.

2.2 Probleemi aktuaalsus

Probleem on aktuaalne, kuna broneeringusüsteemi puudumine raskendab mõlema osapoole toiminguid. Samuti on veebipõhised broneeringusüsteemid saamas tänapäevaseks standardiks ning nende puudumine võib nii mõnegi kliendi panna mõtlema mõne teise hambaravikeskuse kasuks. Klientidel peaks alati olema võimalus teha broneeringuid iseseisvalt, isegi kui mõned neist sellist käitumist ei eelista ning soovivad ka broneeringusüsteemi olemasolul jätkata broneeringute tegemist vanaviisi.

2.3 Võimalikud lahendused

Broneeringusüsteemi puudumine on probleem ning lõputöö käigus soovitakse leida probleemiga tegelemiseks parim lahendus. Selleni jõudmiseks on oluline osa analüüsil, mille eesmärgiks on sõnastada lahti ülesande olemus ning vaadata sellele otsa võimalikult paljude nurkade alt. Selline käitumine annab suurima tõenäosuse näha ära kõik võimalikud lahendused, et siis valida välja nende seast parim.

Kohila Meedik OÜ registratuuris on hambaravi igapäevatöö korraldamiseks kasutusel sellele spetsialiseeruv programm Dentas [1]. Lisaks paljule muule, mida Dentas hambaravikeskusele võimaldab, on lõputöö raames oluline, et antud programm juba tegeleb nii teenuste, arstide kui ka töögraafikutega. Lõputöö raames ei omata kontrolli hambaravikeskuses kasutusel oleva sisemise süsteemi üle ning ollakse sunnitud tegema mitmed broneeringusüsteemi otsused vastavalt Dentas hambaravisüsteemi võimekusele.

Lihtsaima lahendusena oleks võimalik lõputöö raames hambaravikeskuses kasutusel olevat tarkvara ignoreerida ning vaadelda loodavat broneeringusüsteemi kui eraldiseisvat komponenti. Administraatori õigustega kasutajal oleks võimalik sisestada broneeringusüsteemi erinevaid teenuseid ja arste ning määrata ka nende töögraafikuid. Tavakasutajad saaksid teha sisestatud andmete põhjal broneeringuid. Tehtud broneeringud tuleksid administraatorkasutajale süsteemis nähtavaks ning tal oleks võimalik need manuaalselt kanda üle sisemiselt kasutusel olevasse (st Dentas) hambaravisüsteemi. Kui tavakasutaja seisukohast oleks antud lahendus mugav, siis hambaravikeskusele tooks see kaasa mitmeid probleeme.

Eraldiseisva rakenduse puhul peaks registratuur tegema topelt tööd ning veenduma pidevalt, et nii nende sisemine süsteem kui ka broneeringute rakendus oleksid omavahel kooskõlas. Selliselt käitudes tekiks tahes tahtmata rohkem probleeme kui kasu. Inimeste broneeringuid tuleks lükata pidevalt tagasi, kuna broneeringusüsteemis kuvatud ajad ei kattunud sisemise süsteemiga. Samuti oleks pidevalt vajalik hoida paralleelselt töötamas kahte süsteemi, mis tegelevad väga sarnaste ülesannetega. Eraldiseisev lahendus ei tee mugavamaks registratuuri tööd ning ei ole seetõttu sobiv.

Analüüsi käigus on jõutud selgusele, et rakendus suudab pakkuda hambaravikeskusele väärtust ainult juhul, kui ta on kasutuseloleva sisemise programmiga integreeritud. Uurides programm Dentase integreerimisvõimalusi, selgub, et Dentas pakub online broneeringute tegemiseks nii broneeringute liidest kui ka veebirakendust, mis eelnimetatud broneeringuliidest kasutab. Järgnevalt analüüsitakse olemasolevat veebirakendust ning tuuakse välja, miks ei soovita lõputöö raames seda kasutada.

Olemasoleva lahenduse disain on algeline ning ei vasta seetõttu ka hambaravikeskuse ootustele (Lisa 2). Samuti on broneerimisprotsess koondatud ühte vaatesse, mis tekitab mitmeid probleeme. Selle asemel, et lasta kliendil valida nii teenus, arst, broneeritav aeg

kui ka sisestada oma isikuandmed kõik ühes vaates, oleks eelistatum jagada need protsessid eraldi sammudeks. Kasutajale on mugavam, kui ta saab korraga tegeleda vaid ühe toiminguga. Eriti suurel määral parandaks selline lähenemine kasutuskogemust väikestel ekraanidel, näiteks mobiilivaadetes, kus on keeruline suurt hulka infot korraga kuvada.

2.4 Loodava lahenduse skoop

Dentase poolt pakutav veebirakendus ei ole Kohila Meedik OÜ-le sobiv lahendus. Lõputöö eesmärgiks võetakse luua sellele alternatiivne veebirakendus, mis sarnaselt olemasolevale tegeleb broneeringutega Dentase broneeringuliidese kaudu. Lahendus pakub maksimaalset võimalikku mugavust hambaravikeskusele, olles otseses suhtluses sisemiselt kasutusel oleva süsteemiga, võimaldades samal ajal lõputöö autoril disainida veebirakendust kasutajasõbralikumalt, arvestades Kohila Meedik OÜ hambaravikeskuse broneeringute vajadusi.

Broneeringusüsteemi loomisel arvestatakse, et klientidele peab säilima võimekus teha broneeringuid vanaviisi. Isegi kui loodav broneeringusüsteem pakub klientidele võimekust teha broneeringuid mugavaimal viisil, leidub alati kliente, kelle jaoks muutustega harjumine on ebamugav ning kes soovivad teha toiminguid vastavalt sellele, kuidas nad on harjunud neid tegema. Lõputöö raames loodav broneeringusüsteem peab pakkuma klientidele lisavõimekust broneeringute tegemiseks, mitte asendama praegused variandid.

Loodav prototüüp saab olema veebirakendus, kuna selline lahendus pakub klientidele maksimaalset mugavust teha broneeringuid neile sobival ajal ja keskkonnas. Veebirakendust on võimalik klientidel kasutada nii arvutis kui ka telefonis.

Hambaravikeskusel Kohila Meedik OÜ on olemas ka interneti koduleht [2], millega broneeringusüsteem saaks olla seotud, kuid lõputöö ajaliste piirangute tõttu käsitletakse loodavat broneeringusüsteemi prototüüpi kui sellest eraldi seisvat komponenti. Et broneeringusüsteem oleks kasutajale leitav, peaks see ideaaltingimustes olema kas olemasolevalt kodulehelt leitav või tuleks laiendada loodavat broneeringusüsteemi olema ühtlasi ka hambaravikeskuse koduleht.

3 Loodava rakenduse analüüs

Järgnevalt pannakse paika prototüübi funktsionaalsed ja mittefunktsionaalsed nõuded. Samuti analüüsitakse prototüübi loomiseks vajalikke tehnoloogiad ning valitakse nende seast välja lõputöö jaoks sobivamad. Rakendusele esitatud funktsionaalsed ja mittefunktsionaalsed nõuded on kooskõlastatud hambaravikeskusega.

3.1 Funktsionaalsed nõuded

Rakendusele esitatakse kliendipoolsed funktsionaalsed nõuded:

- Soovin näha broneeritavaid teenuseid.
- Soovin näha kõikide teenuste arste.
- Soovin näha teenuse jaoks broneeritavaid aegu.
- Soovin filtreerida teenuse broneeritavaid aegu arstide kaupa.
- Soovin filtreerida teenuse broneeritavaid aegu päevade kaupa.
- Soovin filtreerida teenuse broneeritavaid aegu määratud ajavahemiku alusel.
- Soovin esitada broneeringuid valitud ajale.
- Soovin sisestada broneeringuga seotud isikuandmed.
- Soovin broneeringu andmetesse jätta vajadusel täpsustava sõnumi.
- Soovin saada broneeringu sooritamisel teavitust e-mailile.
- Soovin saada broneeringu sooritamisel teavitust telefonile.
- Soovin näha esitatud broneeringu ülevaadet.
- Soovin esitatud broneeringut tühistada.
- Soovin broneeringu ülevaate juures näha, kui minu broneering on tühistatud.

3.2 Mittefunktsionaalsed nõuded

Rakendusele esitatakse kliendipoolsed mittefunktsionaalsed nõuded:

- Soovin, et rakendus oleks eestikeelne.
- Soovin, et rakendus oleks kasutatav arvutis.
- Soovin, et rakendus oleks kasutatav telefonis.

Rakendusele esitatakse hambaravikeskuse poolsed mittefunktsionaalsed nõuded:

- Soovin, et kliendile kuvatud teenused oleksid vastavuses andmetega sisemises süsteemis.
- Soovin, et kliendile kuvatud arstid oleksid vastavuses andmetega sisemises süsteemis.
- Soovin, et kliendile kuvatud broneeritavad ajad oleksid vastavuses andmetega sisemises süsteemis.
- Soovin, et kliendi broneering jõuaks sisemises süsteemis arsti töögraafikusse.
- Soovin, et kliendi broneeringut oleks sisemise süsteemi kaudu võimalik tühistada. Tühistamine peaks olema kajastatud ka kliendile.
- Soovin, et broneeringusüsteem ei lubaks teha kliendil broneeringuid ilma isikutuvastuseta. Teisisõnu soovin, et broneeringu puhul vastaksid isikuandmed alati konkreetsele inimesele.

3.3 Sisemise süsteemiga integreerimine

Dentase koduleht kirjeldab „Ühilduvus ja Integratsioonid“ aknas ühe oma integratsioonivõimekusena online broneeringute korraldamist. Kodulehel puudub aga täpsem ülevaade, mida on selle all mõeldud.

Et saada broneeringute integreerimise võimekusest parem ülevaade, võeti lõputöö raames ühendust Dentase kasutajatoega. Selgus, et pakutakse nagu ka eelnevalt juba mainitud, võimekust kasutada nii valmis olevat veebirakendust kui ka broneeringute liidest eraldi.

Kasutajatoega kontakteerudes edastati lõputöö autorile broneeringuliidese dokumentatsioon. Lisaks selgus, et arenduse faasis on võimalik broneeringuliidest kasutada vastu demo kliinikut. Demo kliiniku kasutamine lihtsustab arenduse protsessi, võimaldades testida kõiki rakenduse nõudeid, ilma et süsteem peaks arenduse faasis olema seotud päris andmetega, päris kliinikuga. Demo kliinikut hoitakse varustatud testandmetega, mis elimineerib ka vajaduse koostada andmete küsimiseks mingisugune kogus testandmeid.

Veendumaks, et läbi arendatava prototüübi tehtud broneeringud jõuavad Dentase süsteemi arstide töögraafikusse nähtavale, edastati lõputöö autorile ka arstina demo

kliinikusse sisse logimise õigused läbi Dentase programmi. Arsti õigustega sisse logimine on piisav, et veenduda lõputöö raames arendatava broneeringusüsteemi töös.

Broneeringusüsteemi korrektsele integreerimisele aitab kaasa Dentase poolt pakutav online broneeringutega töötamise juhend [3].

Lõputöö raames toimub broneeringusüsteemi prototüübi arendus vaid vastu demo kliinikut. Lõputöö fookusesse ei jää broneeringusüsteemi sidumine Kohila Meedik OÜ kliinikuga.

3.3.1 Dentase broneeringuliidese sobilikkus

Broneeringuliidese dokumentatsioon kirjeldab lõputöö raames oluliste toimingutena järgmist:

- Kliiniku üldandmete lugemine (muuhulgas kliiniku teenused ja arstid).
- Teenuse broneeritavate aegade lugemine.
- Aja broneerimine.
- Broneeritud aja lugemine.
- Broneeritud aja tühistamine.

E-maili ning telefoni teavitused saadetakse läbi broneeringuliidese tehtud broneeringute puhul kliendile automaatselt. Teavituste struktuuri on võimalik Dentase tarkvaras vastavalt ettevõtte vajadusele konfigureerida, kasutades soovi korral ka broneeringus sisalduvaid isikuandmeid. Lõputöö raames ajaliste piirangute tõttu aga teavituste struktuuri ei konfigureerita ning seetõttu ka seda tegevust eraldi ei dokumenteerita.

Broneeringuliides on piisav, et täita kõik broneeringusüsteemile esitatud nõuded.

3.4 Autentimine

Broneerimissüsteemi puhul on registratuurile oluline, et broneering oleks seotud kindla isikuga. Broneeringuga seotud isiku puhul on oluline tema isikukood, nimi, e-maili aadress ja telefoninumber.

Lihtsaim lahendus oleks lasta kasutajatel need andmed broneeringute puhul iga kord manuaalselt sisestada. See ei ole aga optimaalne mitmel põhjusel.

Esiteks on see ebamugav lähtudes kasutajakogemusest. Lastes kasutajal broneerimisprotsessis sisestada manuaalselt kõik vajalikud isikuandmed, on ta kohustatud tegelema suure hulga andmete sisestamisega. Lisaks tekib oht, et kas tahtlikult või mitte, teeb inimene isikukoodi või nime sisestades vea. Ebakorrektsete andmete puhul poleks registratuuril võimalik broneeringut konkreetse inimesega seostada ning kui broneeringusüsteem sellist tegevust aktsepteeriks, tuleks registratuuril pidevalt selle probleemiga tegeleda.

Lisaks võimaldaks autentimise puudumine pahatahtlikel kasutajatel teha järjest suvalisi broneeringuid kõikidele broneeritavatele aegadele. Autentimissüsteemi olemasolul on selline käitumine samuti võimalik, kuid vähem soodustatud, kuna inimene peab broneeringute tegemiseks olema autenditud. Autentimise tulemusena on võimalik broneeringud isikuga siduda, mis vähendab kasutajate motiveeritust pahatahtlikule tegevusele.

Autentimissüsteem lihtsustaks isikuandmete sisestamise protsessi, teeks raskemaks pahatahtliku käitumise ning annaks ka kasutajatele endile suurema kindluse, et sisestatud broneeringuandmed on korrektsed.

Autentimist broneeringusüsteemi implementeerides on vajalik veenduda, et kasutajal puudub võimekus autentimisest mööda hiilida. Kuna lõputöö raames kasutatav broneeringuliides võimaldab teha broneeringuid mis iganes isikuandmetega, on loodava broneeringusüsteemi juures oluline, et kasutajal puuduks ligipääs broneeringuliidese kaudu tehtud päringutele. Päringuid lugedes oleks võimalik kasutajatel koostada analoogseid päringuid iseseisvalt, hiilides seeläbi mööda autentimise nõudest, mille eest tagarakendus peab vastutama.

3.4.1 Autentimisteenuse valik

Eestis on autentimiseks kasutusel peamiselt 3 erinevat viisi:

- ID-kaardiga autentimine.
- Smart-ID autentimine.
- Mobiil-ID autentimine.

Ideaaltingimustel on vajalik tagada kasutajatele võimekus autentida end mis iganes talle sobivamail viisil. Lõputöö ajaliste piirangute tõttu implementeeritakse neist aga prototüübi raames vaid üks.

Lõputöö raames implementeeritakse autentimine läbi Smart-ID teenuse. Teenuse valikul lähtuti nii kasutusmugavusest, kasutuselevõtu kergusest kui ka lõputöö autori varasemasest teenustega kokkupuutest. ID-kaardi puhul oleks isikutuvastus piiratud kaardilugeja olemasoluga ning seetõttu ei oleks kasutajatele mugav. Mobiil-ID ja Smart-ID valikul eelistatakse lõputöö raames Smart-ID teenust, kuna lõputöö autoril eksisteerib sellega varasem kasutuskogemus. Samuti on Smart-ID teenuse integreerimise protsess arendajate jaoks suurepäraselt dokumenteeritud.

3.4.2 Smart-ID integreerimine

Nagu Smart-ID integreerimisprotsess seda kirjeldab, käib e-teenusele Smart-ID toe lisamine kiirelt ja lihtsalt [4]. Integreerimisprotsessi mugavdamiseks ning teenuse testimiseks on olemas testkontod ja testteenused, mis võimaldavad enne lepingu sõlmimist arendada lahendus turvaliselt lõpuni. Lõputöö raames simuleeritakse autentimisi vaid vastu Smart-ID testkeskkonda.

Smart-ID toe lisamise protsessi kirjelduses on toodud välja erinevad avalikud vabavaralised tarkvarateegid, mida kõiki on võimalik oma rakenduses kasutada ning mille eesmärk on lihtsustada Smart-ID päringute tegemist. Ametlikud Smart-ID teegid on saadaval PHP ning Java keelele, kuid leidub ka kolmanda osapoole teeke Ruby, .NET, Go ja Rust keskkondadele [5].

3.5 Tehnoloogia valik

Lõputöö raames arendatav prototüüp on jaotatud esi- ning tagarakenduseks. Prototüüp oleks võimeline toimima ka ilma tagarakenduseta, kuid sellisel lähenemisel oleksid mitmed miinused. Esirakenduse puhul on oluline, et broneeringutega seotud päringuid oleks võimalik teha võimalikult mugaval kujul. Lõputöö prototüübi raames on otsustatud ehitada esirakenduse ning broneeringuliidese vahele tagarakenduse näol vahekiht, mis võimaldab esirakendusel teha päringuid talle sobival kujul, olles samal ajal sõltumatu broneeringuliidese ülesehitusest.

Nii esi- kui ka tagarakenduse puhul on oluliseks osaks tehnoloogia valik. Järgnevalt analüüsitakse erinevaid tehnoloogiaid, lähtudes nii nende sobivusest kui ka lõputöö autori kogemusest. Parima tehnoloogia valiku tegemiseks selgitatakse ka lühidalt sellele esitatud nõudeid, mille täitmiseks peab valitud tehnoloogia olema piisav.

3.5.1 Esirakenduse tehnoloogia valik

Esirakenduse eesmärgiks on pakkuda kasutajatele meediumit rakenduse kasutamiseks. Esirakendus peab olema koheselt lihtne ja arusaadav ka inimestele, kel on sellega esimene kokkupuude. Inimeste arvamus millegi üle kujuneb ruttu ning on väga suurel määral seotud esmamuljega. See kehtib ka veebilehtede puhul, kus ei tohi seetõttu kunagi eeldada, et kasutajatel on aega rakendusega tutvuda ning seda tundma õppida. Kasutajale on oluline presenteerida korraga vaid nende jaoks olulist infot ning teha seda viisil, mis on neile üheselt arusaadav.

Kuna esirakendus on oma olemuselt tavapärane veebirakendus, oleks see võimalik luua mistahes JavaScript raamistikuga. Erinevaid raamistikke analüüsides oleks võimalik tuua poolt- ja vastu argumente neile kõigile. Lõputöö raames soovitakse minimaliseerida aega, mis kulub uute tehnoloogiate õppimisele. Samuti soovitakse vähendada tõenäosust sattuda pidevalt ootamatute probleemide otsa. Seetõttu on vägagi eelistatud tehnoloogiad, millega on lõputöö autoril olnud varasem kokkupuude.

Tulenevalt lõputöö autori varasemasest kogemusest, valitakse lõputöö esirakenduse raamistikuks React [7]. React on suuteline täitma kõik esirakendusele esitatud nõuded, võimaldades samal ajal kasutada ka mugavalt mitmeid erinevaid komponenteeke.

3.5.2 React komponentteegi valik

Esirakenduse arendusel on ajaliste piirangute tõttu oluline lihtsustada erinevate UI (*User Interface*) elementide loomise protsessi. Protsessi kiirendamiseks on otsustatud kasutada mõnda React raamistikuga ühilduvat komponentteeki. Komponentteegid pakuvad arendajale kasutuseks laialdaselt levinud UI elemente nagu näiteks sisendi väljad, nupud, kalendrid, menüü aknad ja muud. Järgnevalt analüüsitakse mõnda React komponentteeki ning tehakse nende seast valik.

Ant Design on üks populaarsemaid React raamistiku komponenditeeke, eriti ärirakenduste loomisel [8]. Ant Design pakub laia valikut komponente. Raamistik on

raskekaaluline ning mõeldud peamiselt kasutuseks suuremates rakendustes ning ei ole seega lõputöö raames sobilik.

Chakra UI on komponentide teek, mis keskendub lihtsusele, paindlikkusele ja arenduse kiirusele [9]. Nagu ka Ant Design, sisaldab Chakra UI laialdaselt komponente, mis on kõik vastavalt vajadusele kohandatavad. Chakra UI on eriti populaarne arendajate seas, kes otsivad kergekaalulist ja ligipääsetavat lahendust. Võrreldes MUI teegiga [10], pakub Chakra UI aga vähem eeldefineeritud elemente ning ei ole seetõttu valikuna eelistatud.

MUI (Material UI) on komponentide teek, mis põhineb Google Material Design standardil [11]. Raamistiku plussiks on lai kasutajaskond, rohke elementide valik ja kõrge kvaliteet: MUI järgib Google Material Design juhiseid, mis tähendab, et teegi pakutavad komponendid on lisaks ilusale väljanägemisele ka kasutajasõbralikud ning standardiseeritud.

Lõputöö raames on otsustatud kasutada komponentteegina MUI teeki.

3.5.3 Tagarakenduse tehnoloogia valik

Tagarakenduse eesmärk on korraldada broneeringutega seonduvaid toiminguid. Tagarakendus peab olema efektiivne päringute tegemisel ning peab vastutama autentimise toimingute möödapääsematuse ja turvalisuse eest.

Sarnaselt esirakendusele, lähtutakse ka tagarakenduse valikul suurel määral varasemale kogemusele. Lisaks on tagarakenduse raamistiku valikul oluline, et valitud raamistikule eksisteeriks ka Smart-ID autentimise teek.

Lõputöö autoril on tagarakenduste arendusega olnud varasemalt peamine kokkupuude .NET keskkonnas [12]. .NET raamistik võimaldab teha kõiki toiminguid, mille eest tagarakendus peab prototüübi raames vastutama ning lisaks on keskkonnale saadaval ka Smart-ID autentimiseks kolmanda osapoole teek, mis arenduse faasi suurel määral lihtsustab.

Lõputöö raames arendatakse tagarakendust .NET keskkonnas.

3.5.4 Versioonihalduskeskkonna valik

Rakenduse arendusfaasis soovitakse hoida arendatavat koodi mõnes versioonihalduskeskkonnas. Versioonihalduskeskkonnad pakuvad suurimat väärtust, kui projekti arendatakse paralleelselt mitme arendaja poolt, kuid eksisteerib ka väärtusi, mis esinevad individuaalsel tasemel. Versioonihalduskeskkonnad võimaldavad arendajatel teha süsteemis muutusi, tagades alati võimekuse minna tagasi enne muutuste tegemist olevasse koodibaasi seisu. Lisaks pakub see arendajale koodibaasi salvestuste näol head ülevaadet tehtud tööst ning võimaldab töötada projekti kallal mugavalt erinevates seadmetes, juhul kui selleks peaks olema vajadus.

Lõputöö käigus tehakse versioonihalduskeskkonna valik GitHub-i [13], GitLab-i [14] ning BitBucket-i [15] seast. Eelnimetatud versioonihalduskeskkonnad on valdkonna populaarseimad ning seetõttu valiku tegemisel aktuaalseimad.

Prototüüpi arendatakse lõputöö raames individuaalselt ning seega on keskkonnale esitatud nõuded minimaalsed. Kõik eelnimetatud keskkonnad võimaldavad teha kõiki lõputöö raames olulisi toiminguid.

Lõputöö käigus teostatakse versioonihaldust GitHub keskkonnas, kuna keskkond on valdkonna suurim ning samuti on lõputöö autoril olnud sellega varasemalt suurim kokkupuude.

3.6 Veebirakenduse arhitektuur

Veebirakendust disainides on suur osakaal veebirakenduse arhitektuuril. Hästi ülesehitatud rakendus on kergelt muudetav ning seeläbi paindlik arenduse faasis ette tulevatele muutustele.

3.6.1 Tagarakenduse arhitektuur

Tagarakendus on jaotatud osadeks. Rakenduse osadeks jaotamine parandab koodibaasi loetavust ning lihtsustab seeläbi ka arenduse protsessi.

Tagarakenduse osad on järgnevad:

- Web – koosneb peamiselt kontrolleritest, mis pakuvad esirakendusele võimekust suhelda loodava tagarakendusega.
- DTO – hoiustab rakenduses kasutatavaid andmeedastusobjekte.
- BLL – hoiustab broneeringute tegemisega seonduvat loogikat.

3.6.2 Esirakenduse arhitektuur

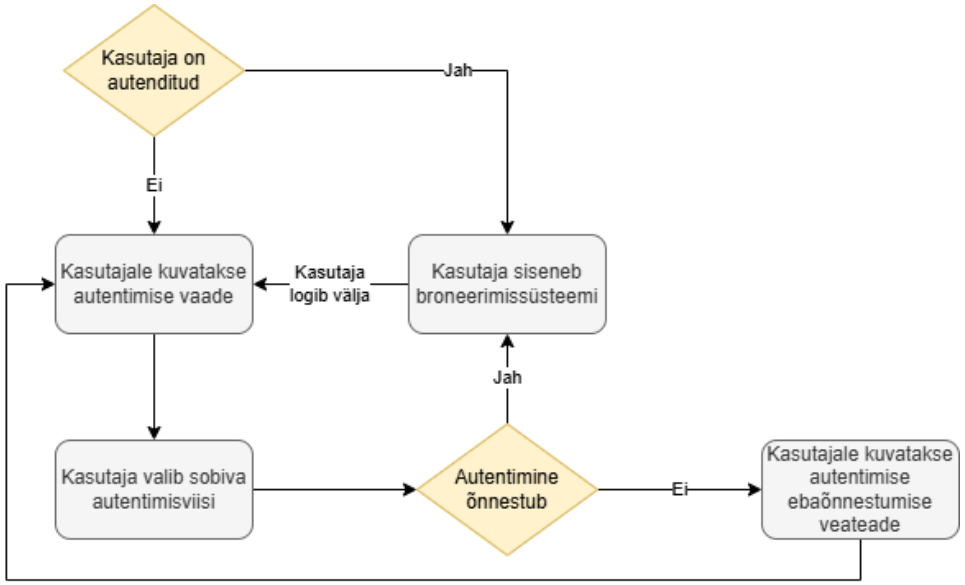
React raamistik on oma olemuselt paindlik (*unopinionated*) ehk teisisõnu lubab see arendajal struktureerida projekte piiranguteta. Piirangute vabadusele saab tuua nii poolt- kui vastu argumente, kuid lõputöö autor peab seda prototüübi arenduse raames pigem eeliseks, eriti veel olukorras, kus prototüüpi arendatakse individuaalselt.

Esirakendus on struktureeritud alljärgnevalt:

- Components – hoiustab React komponente, millest moodustatakse kokku kõik kasutajale nähtavad rakenduse vaated.
- DTO – hoiustab sarnaselt tagarakendusele struktureeritul kujul andmeteedastusobjekte.
- Services – hoiustab teenust päringute tegemiseks vastu tagarakendust. Kasutatakse komponentides, mis peavad vaate kokku panemiseks küsima andmeid läbi tagarakenduse.
- Style – hoiustab rakenduse disainiga seotud CSS (*Cascading Style Sheets*) faile. CSS failid on veebirakenduste elementide stiili kirjeldavad failid, mille abil kujundatakse veebilehtede vaateid [16].
- Utils – hoiustab abistavaid teenuseid. Kausta eesmärk on koondada kokku korduv kasutatav loogika, mis seeläbi vähendab koodi duplikatsiooni ja parandab loetavust.

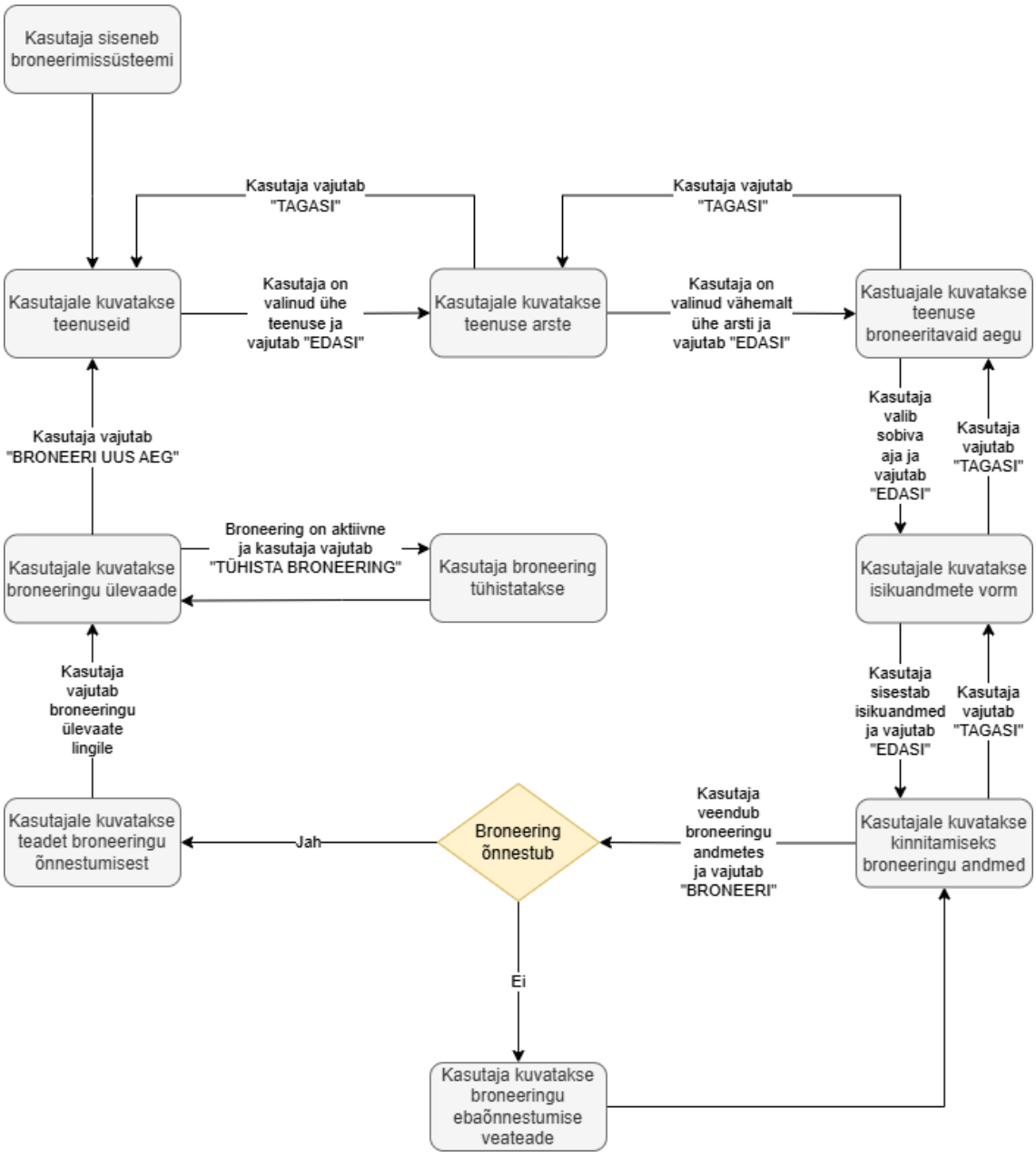
3.7 Veebirakenduse disain

Veebirakendus peab oma disaini poolest pakkuma kasutajatele mugavamat broneerimisprotsessi kui seda võimaldab olemasolev lahendus. Analüüsi käigus on selgunud, et soovitakse broneerimisprotsess jagada loogilisteks sammudeks. Samm-haaval broneerimine lihtsustab nii vaadete kujundamist kui ka broneerimise protsessi kasutajale. Järgnevalt selgitatakse veebirakenduse toimingute loogikat illustreerivate jooniste abil. Esmalt käsitletakse autentimise toimingut (Joonis 1).



Joonis 1. Kasutaja autentimise plokkskeem.

Olles korrektselt autenditud, on kasutajal võimalik hakata kasutama broneeringusüsteemi (Joonis 2).



Joonis 2. Broneeringusüsteemi kasutamise plokkskeem.

Esmalt kuvatakse kasutajale broneeritavad teenused. Teenustest on kasutajal võimalik valida välja üks, peale mida on tal võimalik liikuda arstide valiku vaatesse. Arste võib samale teenusele olla mitu ning kasutajal tuleb valida nende seast välja vähemalt üks. Liikudes edasi broneeringuaja vaatesse, võimaldatakse kasutajal filtreerida aegu nii päeva kui ka ajavahemiku alusel. Kui sobiv aeg on valitud, suunatakse klient sisestama oma kontaktandmed, milleks on e-mail ja telefon. Nii nimi kui ka isikukood on juba varem autentimise käigus tuvastatud ning eraldi sisestust ei vaja. Lisaks pakutakse kasutajale võimalust lisada broneeringu andmetesse vajadusel täpsustav kommentaar. Viimaks kuvatakse kasutajale veendumiseks uuesti üle kõik tema broneeringuga seotud andmed.

Olles nendes veendunud, vajutab klient „BRONEERI“, mille peale kuvab broneeringusüsteem kasutajale teadet broneeringu õnnestumisest koos lingiga broneeringu ülevaatele. Kui broneerimise käigus kasutaja autentimine aegub, suunatakse klient end taas autentima.

Kasutades broneeringul saadud unikaalset broneeringu ülevaate linki näeb kasutaja oma broneeringuga seotud infot. Broneeringu infos sisaldub ka selle staatus, mis on kas „Aktiivne“, „Tühistatud“ või „Aegunud“. Lisaks võimaldab vaade broneeringut tühistada.

Broneeringu ülevaate nägemiseks ei pea kasutaja olema autenditud. Eduka broneerimise korral genereeritakse kliendile unikaalse koodiga veebilink, mistõttu on see avalik ainult broneeringu tegijale.

Broneeringu ülevaade hoitakse eraldi aadressil, et sellele oleks võimalik ligi saada ka hiljem ning samuti oleks võimalik see lisada nii e-maili kui ka telefoni teavitustesse, mis Dentase poolt automaatselt välja saadetakse. Aga nagu ka eelnevalt mainitud, seda protsessi lõputöö raames ei dokumenteerita.

Veebirakenduse vaated on toodud välja esirakenduse arenduse peatükis.

4 Rakenduse arendus

Lõputöö raames loodava prototüübi puhul tegeletakse nii esi- kui ka tagarakenduse arendusega. Järgnevalt kirjeldatakse nii esi- kui ka tagarakenduse arendust eraldi peatükkidena.

4.1 Tagarakenduse arendus

Tagarakenduse peamine eesmärk on korraldada broneeringutega seotud toiminguid ja pakkuda võimekust esirakendusele sellega suhtluseks. Tagarakendust arendatakse .NET raamistikus, C# keeles. Arendusel on kasutatud .NET 8.0 versiooni, kuna tegu on lõputöö arendamise hetkel värskeima .NET LTS (*Long Term Support*) versiooniga. Järgnevalt kirjeldatakse põhjendatult erinevaid tagarakenduse arendusel tehtud otsuseid.

4.1.1 Kontrollerite lõpp-punktide disain

Tagarakendus peab pakkuma esirakendusele suhtlust mugavaimal kujul. Teisisõnu peavad tagarakenduse kontrollerite lõpp-punktid olema üles ehitatud vastavalt esirakenduse vajadustele, mitte lähtuvalt kasutatava broneeringuliidese struktuurist. Järgnevalt tuuakse välja lõputöö käigus loodud tagarakenduse lõpp-punktid (Tabel 1).

Tabel 1. Tagarakenduse lõpp-punktid.

AvailableSlotsController		
Teenuse broneeritavate aegade pärimine, vajalik spetsifitseerida teenus ja vähemalt üks arst	GET	*/AvailableSlots/Get?serviceId= &doctorIds=*
BookingController		
Broneeringu tegemine	POST	*/Booking/Make*
Broneeringu andmete pärimine	GET	*/Booking/{bookingCode}/Get*
Broneeringu tühistamine	POST	*/Booking/{bookingCode}/Cancel*
ClinicController		
Kliiniku andmete pärimine	GET	*/Clinic/Get*
DoctorsController		
Teenusega tegelevate arstide pärimine	GET	*/Doctors/Get?serviceId=*
ServicesController		
Teenuste pärimine	GET	*/Services/Get*
SessionController		
Autentimisandmete lugemine	GET	*/Session/GetPersonalDetails*
Autentimisandmete kustutamine	POST	*/Session/ClearPersonalDetails*
SmartIdAuthController		
Smart-ID autentimise alustamine. Tagastatakse kontrollkood ja räsi.	GET	*/SmartIdAuth/BeginAuth*
Smart-ID autentimine. Sisendiks nii autentimise alustamise räsi kui ka isikukood. Prototüübi raames isikukoodi ignoreeritakse ning autentimise protsess simuleeritakse vastu testkeskkonna andmeid.	POST	*/SmartIdAuth/PerformAuth/{personalCode}*

Lõpp-punktide tabel ei kirjelda tabeli loetavuse huvides *Response Body* ega *Request Body* struktuuri.

4.1.2 Dentase broneeringuliidese analüüs

Dentase poolt pakutav broneeringute liides on REST API liides. Päringud teostatakse POST meetodil ning andmed edastatakse ja tagastatakse JSON (JavaScript Object

Notation) formaadis. JSON on JavaScript-ist inspireeritud võtme ja väärtuse paaridest koosnev andmeformaat, mis on veebiteenustes väga laialdaselt kasutusel [17].

Dentase broneeringuliidese dokumentatsioon kirjeldab avalikke lõpp-punkte, mille alusel on võimalik teha sisemise süsteemiga integreeritud broneeringute toiminguid. Päringute sisendite struktuur on muutuv sõltuvalt päringust, kuid eksisteerib komponente, mis peavad olema kaasas igal päringul:

- KEY – Dentas programmist genereeritud kliiniku kood. Kirjeldab, millise kliiniku andmete vastu päring tehakse. Nagu ka eelnevalt kirjeldatud, tehakse arenduse faasis päringuid vastu demo kliinikut ning seega on prototüübi tasemel kliiniku võti vastav demo kliiniku võtmele.

- LANG – kasutatav keel ISO 639-1 formaadis [18]. Dentase süsteemis sisalduvaid andmeid on võimalik küsida erinevates keeltes. Kui näiteks pakutaval teenusel on lisaks eestikeelsele nimele defineeritud ka inglisekeelne teenuse nimi, on broneeringuliides võimeline selle tagastama inglise keeles defineeritud kujul. Lõputöö raames ajaliste piirangute tõttu eri keelte funktsionaalsustega ei tegeleta ning kasutatakse kõikide päringute puhul eesti keelt.

Järgnevalt kirjeldatakse Dentase broneeringuliidese sisendite (Joonis 3) ja väljundite (Joonis 4) struktuuri.

```
{
  “param“: SISEND_JSON_OBJEKT_STRING_TÜÜBIKS_TEISENDATUNA
}
```

Joonis 3. Dentase broneeringuliidese päringu sisendi struktuur.

```
{
  “d“: {
    “Value“: VÄÄRTUS,
    “Data“: ANDMED,
    “ErrorCode“: VEAKOOD,
    “ErrorMessage“: VEATEADE
  }
}
```

Joonis 4. Dentase broneeringuliidese päringu väljundi struktuur.

Päringu vastuses leitav objekt *d* sisaldab järgmisi andmeid:

- Value – kasutusel, kui tagastatakse mingisugune väiksema struktuuriga väärtus.
- Data – kasutusel, kui tagastatakse suuremaid andmehulki.
- ErrorCode – veakood, 0, kui viga ei eksisteeri.
- ErrorMessage – vea sõnum, tühi, kui viga ei eksisteeri.

Arenduse faasis on loodud päringutega tegelemiseks eraldi komponent, mis suure osa päringute esitamise ning vastuse tõlgendamise loogikast enda sisse ära peidab. Kui päringute sisendite ja vastuste väline struktuur on alati ühesugune, saab seda eeldada ning lasta päringutega tegeleval moodulil sellega arvestada. Sellisel juhul on tagarakenduse tasemel võimalik päringute sisendeid ja vastuseid struktureerida vastavalt vajadusele ning lasta päringutega tegeleval komponendil teha kõik vahepealsed teisendused automaatselt.

4.1.3 Dentase broneeringuliidese kasutamine

Tagarakenduse lõpp-punktide puhul puudub tihtipeale analoogne Dentase broneeringuliidese päring, mis tagastaks just konkreetse lõpp-punkti jaoks vajaliku info. Teatud juhtudel on võimalik teha ainult mingi üleüldisem päring, mis tagastab andmeid rohkem kui on selles kontekstis oluline.

Näiteks eksisteerib tagarakenduse kontrolleris lõpp-punkt, mille ülesanne on väljastada kõik kliiniku teenused. Dentase broneeringuliidese kaudu pole aga võimalik teha päringut, mis tagastaks vaid kliiniku teenused. Kliiniku teenuseid on võimalik pärida vaid koos kliiniku arstidega. Üks võimalus sellise probleemiga tegelemiseks oleks kujundada tagarakenduse lõpp-punktid vastavalt broneeringuliidese omadele, või elimineerida tagarakenduse vahekiht täielikult ning lasta esirakendusel suhelda broneeringuliidesega vahetult. Ometi aga ei ole broneeringuliidese kasutamine esirakenduses mugav ning samuti ei taheta kujundada tagarakendust vastavalt broneeringuliidese limitatsioonidele. Tagarakenduse lõpp-punktid peavad olema disainitud vastavalt esirakenduse vajadustele ning ainult reaalne implementatsioon peab tegelema broneeringuliidesest tulenevate eripäradega.

Seega eksisteerib süsteemis pidevalt olukordi, kus lõpp-punktide raames optimaalset päringut ei eksisteeri ning vastuse kokku panekuks tuleb läbi broneerimisliidese küsida suurem hulk andmeid.

Probleemi lihtsaima lahendusena oleks võimalik lasta sellisel olukorral süsteemis eksisteerida. Sellisel juhul oleks aga liidese kasutamine ebaefektiivne ning sellist lähenemist ei ole otsustatud lõputöö raames rakendada. Isegi kui vähene päringute optimeerimine ei mõjutaks broneeringusüsteemi finantsilises mõttes, on igal päringul ikkagi oma hind. Antud kontekstis on päringute peamine hind nende peale kuluv aeg, mida soovitakse hoida broneeringusüsteemi raames võimalikult väiksena.

Olukordades, mil broneeringuliidese kaudu on võimalik küsida andmeid vaid suuremal hulgal, on mõistlik ignoreeritud vastuse tulemused hilisemaks kasutuseks mällu salvestada. Sellist protsessi nimetatakse andmete vahemälus hoiustamiseks.

4.1.4 Andmete vahemälus hoiustamise analüüs

Andmete vahemälus hoiustamine (*caching*) on andmete talletamine hilisemaks kasutuseks. Andmete hoiustamise vajalikkust illustreeritakse järgnevalt kirjeldatud olukorraga.

Kui klient alustab broneeringusüsteemis broneerimist, küsitakse läbi tagarakenduse esimesse vaatesse kõik broneeritavad teenused. Tagarakenduse poolel Dentase broneeringuliidese kaudu teenuseid küsides saadakse samal aja infot ka arstide kohta. Tagarakenduse kohustuseks jääb hoiustada nii teenused kui ka arstid, vaatesse saadetakse aga ainult küsitud info ehk antud olukorras teenused.

Järgmise sammuna on kasutaja valinud mõne teenustest ning soovib näha sellele vastavaid arste. Broneeringuliidese puhul jällegi analoogne päring puudub, kuid on võimalik küsida välja kõik teenusele broneeritavad ajad. Broneeringu ajad hoiavad endas informatsiooni, millise arstiga on nad seotud. Vastust töödeldakse ning leitakse seeläbi üles kõik teenuse arstid. Esirakendusse saadetakse välja arstid ning broneeritavate aegadega seonduv info hoiustatakse.

Kui klient on broneeringusüsteemis valinud ka sobivad arstid, kuvatakse talle broneeritavaid aegu. Kuna tagarakendus on eelneva päringu raames jätnud meelde broneeritavate aegadega seonduva info, on võimalik vastus kokku panna hoiustatud andmetest. Kaob ära vajalikkus teha uuesti Dentase broneeringusüsteemi päring ning seeläbi optimeeritakse selle kasutust, kui hiljuti on analoogne päring juba tehtud.

Hoiustamise efektiivsus suureneb veelgi, kui broneeringusüsteemi käitumist vaadelda rohkemate klientide puhul. Kui mitmed kliendil samal ajal tagarakenduse päringuid teevad, suureneb tõenäosus, et vastus on võimalik kokku panna vaid hiljuti hoiustatud andmetest.

Juhtumid, kus mõni broneeritav aeg on juba aegunud, aga klient üritab rakenduse kaudu teha sellele broneeringut, lahendatakse broneeringuliidese poolt. Selline broneering broneeringuliidese kaudu läbi ei lähe ning selle kohta tagastatakse ka veateade, mida kasutajale broneeringusüsteemis kuvatakse.

Lõputöö raames on otsustatud hoiustada broneeringuliidese kaudu tagastatavaid andmeid 10 minuti jooksul. Erinevat tüüpi andmeid oleks võimalik hoiustada erineva ajapikkuse jooksul. Näiteks kliiniku andmeid, mis on vähem muutuvad, võiks hoiustada pikemalt ja broneeritavate aegade andmeid, mis on rohkem muutuvad, hoiustada lühemalt. Lõputöö arenduse raames on aga otsustatud hoida andmete hoiustamine võimalikult lihtsal kujul.

Andmete vahemälus hoiustamine oli esialgselt tingitud tulenevalt Dentase broneeringuliidese päringute struktuurist, kuid suur tähtsus on ka selle muudel eelistel, mida ta veebirakendusele pakub, eriti tulevikus rohkemate kasutajate puhul.

Antud lähenemine võimaldab mugavamat arendust ka esirakenduse poolel, kus peab seeläbi vähem muretsema, kui erinevad React komponendid broneeringuvaadete vahel navigeerides pidevalt kinnituvad ja eemalduvad (*mounting*, *unmounting*). Isegi kui edasi tagasi liikudes tehakse tagarakenduse pihta ikkagi iga kord päring, garanteeritakse vähemalt Dentase broneeringuliidese optimaalne kasutamine.

4.1.5 Andmete vahemälus hoiustamise implementatsioon

Lõputöö raames teostatakse andmete hoiustamist .NET IMemoryCache abil [19].

IMemoryCache vaikimisi implementatsioon võimaldab prototüübi raames kõiki vajalikke toiminguid. IMemoryCache funktsionaalsusi on võimalik rakenduses kasutada täiendades Program.cs konfiguratsiooni (Joonis 5).

```
builder.Services.AddMemoryCache();
builder.Services.AddDistributedMemoryCache();
```

Joonis 5. Andmete vahemälus hoiustamise konfiguratsioon.

Kirjeldatud konfiguratsioon vastutab, et DI (Dependency Injection) konteinerisse registreeritakse IMemoryCache vaikimisi implementatsioon. Kui seejärel mõni rakenduse teenus IMemoryCache tüüpi objekti enda konstruktorisse küsib, sisestab DI sinna registreeritud implementatsiooni.

4.1.6 Autentimise analüüs

Analüüsi käigus selgus, et prototüübi arenduse raames on eesmärgiks implementeerida süsteemi Smart-ID autentimine. Kuna prototüübi tagarakenduse arendus toimub .NET keskkonnas, kasutatakse autentimise lihtsustamiseks vastavat kolmanda osapoole teeki [6]. Teek on kaitstud MIT listsentsiga ning on seetõttu lõputöö raames kasutamiseks sobiv.

Teegi kasutamine rakenduses oleks võimalik mitmel viisil. Naiivseim lahendus oleks broneeringusüsteemi koodibaasi kopeerida autentimisega tegelev osa. Antud käitumine ei oleks aga otstarbekas, kuna see suurendab ebavajalikul määral koodibaasi ning enamasti eksisteerib teekide kasutamiseks mingisugune sobivam viis.

Parem variant oleks lisada teegi repositoorium broneeringusüsteemi repositooriumile alammoodulina (*submodule*) [20]. Alammooduli lisamise asemel eksisteerib aga antud kontekstis eelistatum variant, kuna teek on saadaval ka Nuget paketina [21].

Nuget paketid on .NET ökosüsteemi ühed olulisemad komponendid, mis võimaldavad arendajatel koodi mugavalt jagada. Paketid sisaldavad eelkompileeritud koodi, mis lihtsustab korduvkasutatavate lahenduste integreerimist projektidesse, vähendades arendusaja kulu ja pakkudes standardiseeritud viisi kolmanda osapoole lahenduste kasutamiseks.

Smart-ID .NET teek ei sisalda dokumentatsiooni selle kasutusest. Väliste dokumenteerimata teekide puhul tuleb nende käitumist sageli jälgida läbi testides toimuva. Hästi kirjutatud testid annavad hea ülevaate teegi kasutamisest. Lõputöö raames on rakendatud Smart-ID autentimist vastu testkeskkonda analoogselt teegi testides toimuvale.

Smart-ID pakub võimalust teha arenduse faasis päringuid vastu testkeskkonda, kasutades testkeskkonnas eksisteerivad testkasutajate andmeid [22]. Testkeskkonna kasutajad simuleerivad kasutaja poolset Smart-ID vastuse protsessi. Erinevad testkasutajad on

konfigureeritud tagastama erinevaid vastuseid, mis tagab arenduse faasis võimekuse testida erinevaid Smart-ID vastuse stsenaariume.

4.1.7 Autentimise implementatsioon

Õnnestunud autentimise tulemusena tagastatakse inimese isikukood ning ees- ja perekonnanimi. Broneerimissüsteemi raames on oluline neid andmeid broneerimisprotsessi vältel talletada.

Autentimise andmeid säilitatakse tagarakenduse poolel, sidudes nad .NET raamistiku poolt automaatselt genereeritud sessiooniga, mille kohta koostatakse ka automaatselt veebiküpsis, mis sessiooni identifikaatorit sisaldab [23]. Õnnestunud autentimise korral lisatakse sessiooni küpsis vastusesse.

.NET raamistik muudab sessiooni andmetesse isikuandmete kirjutamise ja sealt nende lugemise väga mugavaks. Sessiooni andmetesse salvestatakse isikuandmeid kontrolleri meetodi tasemel (Joonis 6).

```
HttpContext.Session.SetString(nameof(PersonalDetailsDTO.PersonalCode), authenticationIdentity.IdentityCode);
HttpContext.Session.SetString(nameof(PersonalDetailsDTO.FirstName), authenticationIdentity.GivenName);
HttpContext.Session.SetString(nameof(PersonalDetailsDTO.LastName), authenticationIdentity.Surname);
```

Joonis 6. Sessiooni andmetesse isikuandmete salvestamine.

Sessiooni andmetesse isikuandmete lisamisel koostatakse automaatselt sessiooni küpsis. Esirakenduse vastutada on see küpsis säilitada ning anda see hiljem uuesti broneeringu tegemise päringusse kaasa. Broneeringu tegemise päringu puhul on võimalik vastavalt sessiooni küpsises sisalduvale identifikaatorile korrektse kasutaja isikuandmeid lugeda (Joonis 7) ning kasutada neid broneeringu andmetena. Välja logides on võimalik isikuandmed eemaldada (Joonis 8).

```
string? personalCode = HttpContext.Session.GetString(nameof(PersonalDetailsDTO.PersonalCode));
string? firstName = HttpContext.Session.GetString(nameof(PersonalDetailsDTO.FirstName));
string? lastName = HttpContext.Session.GetString(nameof(PersonalDetailsDTO.LastName));
```

Joonis 7. Sessiooni andmetest isikuandmete lugemine.

```
HttpContext.Session.Remove(nameof(PersonalDetailsDTO.PersonalCode));
HttpContext.Session.Remove(nameof(PersonalDetailsDTO.FirstName));
HttpContext.Session.Remove(nameof(PersonalDetailsDTO.LastName));
```

Joonis 8. Sessiooni andmetest isikuandmete eemaldamine.

4.2 Esirakenduse arendus

Esirakenduse arendamiseks on kasutatud React raamistikku koos TypeScript programmeerimiskeelega [24]. React on populaarne JavaScripti teek, mis võimaldab luua modulaarseid ja dünaamilisi kasutajaliideseid, pakkudes komponentidel põhinevat arendust. Arendusprotsessi käigus on lisaks kasutatud Vite tööriista, mis pakub kiiret arendusserverit ja optimeeritud ehitustööd [25]. Kasutatud Reacti versioon on 18.3.1.

TypeScript on JavaScripti laiendus, mis lisab tugeva tüübisüsteemi. See võimaldab arendajatel määrata, millised andmetüübid igas rakenduse osas lubatud on, vähendades seeläbi vigade teket ja parandades koodi loetavust. TypeScript kompileeritakse tavalisse JavaScripti, mida kasutatakse hiljem brauseris. Kasutatud TypeScripti versioon on 5.6.3.

4.2.1 Suhtlus tagarakendusega

Esirakendus suhtleb tagarakendusega päringute kaudu. Selleks, et päringuid oleks lihtne ja efektiivne teostada, on loodud eraldi fail, mis nendega tegeleb. Kuna tagarakenduses on kasutusel sessiooni küpsis, tuleb päringute tegemisel arvestada, et autentimisel saadud küpsis antakse kaasa broneeringutega seotud päringutele. Kohustuslik on küpsis kaasa anda vaid broneeringu tegemise päringule, kus sessiooni isikuandmeid loetakse, kuid lihtsuse huvides antakse see kaasa kõigile.

Päringuid teostatakse Axios teegi abil [26], mis lihtsustab HTTP päringute [27] tegemist. Päringute haldamiseks luuakse päringutega tegeleva klassi konstruktoris AxiosInstance, mida tõlgendatakse päringute baasina (Joonis 9). Selle instantsi loomisel seadistatakse järgnevad parameetrid:

- Baasaadress, millele kõik päringud suunatakse. Prototüübi tasemel seadistatud *localhost* pordile, kus tagarakendus asub.
- Content-Type, milleks on „application/json“.
- Küpsiste automaatne kaasamine.

```
this.axios = Axios.create({
    baseURL: "https://localhost:7271/",
    headers: {
        common: {
            "Content-Type": "application/json"
        }
    },
    withCredentials: true
});
```

Joonis 9. Esirakenduse päringute konfiguratsioon.

4.2.2 Esirakenduse vaated

Esirakenduse vaated on kujundatud täitma kõiki esirakenduse nõudeid, olles samal ajal kooskõlas veebirakenduse disaini peatükis kirjeldatud broneeringusüsteemi toimingute loogikaga. Järgnevalt tuuakse välja erinevad rakenduse vaated. Vaadetesse küsitud demo kliiniku andmed on privaatsuse huvides asendatud näidisandmetega.

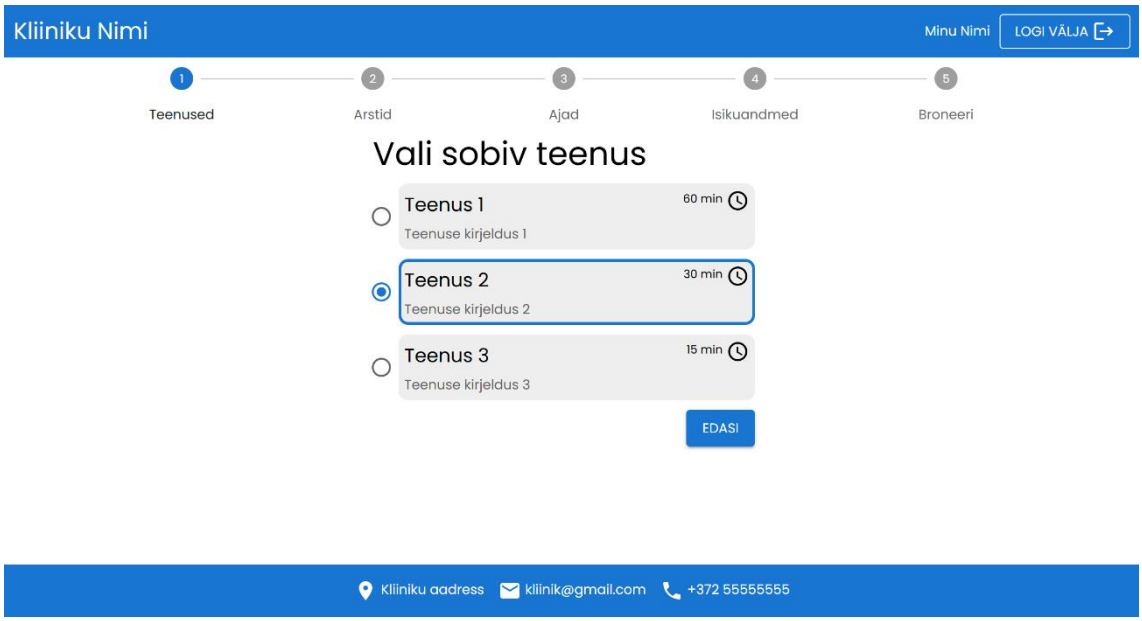
Rakendust kasutades kuvatakse kasutajale esmalt autentimisvaade (Joonis 10). Autentimisvaade võimaldab kasutajal valida nii Smart-ID, Mobiil-ID kui ID-kaardiga autentimise vahel. Lõputöö raames on kujundatud vaated vaid Smart-ID autentimisele. Kasutajale kuvatakse väli isikukoodi sisestamiseks. Kui kasutaja on isikukoodi sisestanud, kuvatakse kontrollkood ning jäädakse ootama Smart-ID vastust (Joonis 11).

Joonis 10. Broneeringusüsteemi esimene autentimisvaade.

Joonis 11. Broneeringusüsteemi teine autentimisvaade.

Kõikides broneeringusüsteemi vaadetes sisaldub lehe päis ja jalus. Päis sisaldab kliiniku nime ning käitub ühtlasi ka lingina baasaadressile. Jalus sisaldab kliiniku kontaktandmeid, mis on interaktiivsed ning viitavad vastavale veebiaadressile. Kliiniku aadressi peale vajutades avaneb sellele vastav *Google Maps* otsing. E-maili ja telefoni aadresside puhul on kasutatud *mailto:* ja *tel:* protokolle [28].

Kui sisselogimine õnnestub, saab kasutaja hakata tegema broneeringut (Joonis 12).



Joonis 12. Broneeringusüsteemi teenuste valimise vaade.

Broneerimine on jagatud viieks erinevaks sammuks ning seda on otsustatud ka illustreerivalt veebilehe üleval osas kuvada. Sammude kuvamine annab kasutajale parema ülevaate broneerimisega seotud toimingutest ning lihtsustab ka nende vahel navigeerimist. Iga sammu puhul kuvatakse kasutajale vaate pealkirjana toimingut, mida ta järgmisele sammule liikumiseks peab tegema. Antud vaate puhul on selleks „Vali sobiv teenus“.

Esimene broneeringusammu vaade illustreerib ka päisesse ilmunud isiku ees- ja perekonnanime ning võimalust logida välja.

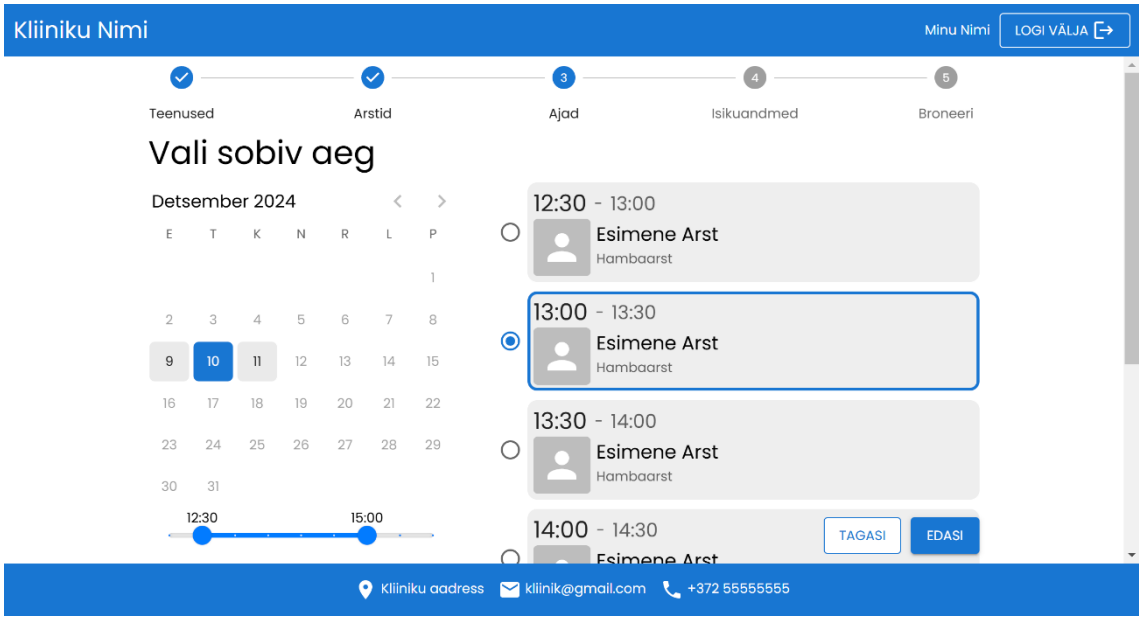
Teenused kuvatakse nimekirjana ning esialgu ei ole neist ükski valitud.

Iga broneeringusammu vaade sisaldab ka sammude vahel navigeerimise nuppe. Esimese sammu puhul tagasi nuppu ei kuvata, viimase sammu puhul asendatakse „EDASI“ tekst „BRONEERI“ tekstiga. Nupud on disainitud olema alati nähtavad. Teisisõnu, isegi kui vaade on keritav, ei kao navigeerimise nupud kunagi jaluse taha peitu.

Liikudes edasi arstide vaatesse (Joonis 13), kuvatakse kõik arstid, kes valitud teenust pakuvad.

Joonis 13. Broneerimissüsteemi arstide valimise vaade.

Vaikimisi on valitud kõik arstid. Kui arstil puudub pilt, asendatakse see vaikimisi pildiga, mida on kuvatud ka joonisel. Olles valinud vähemalt ühe arsti, võimaldatakse navigeerida edasi aegade vaatesse (Joonis 14).



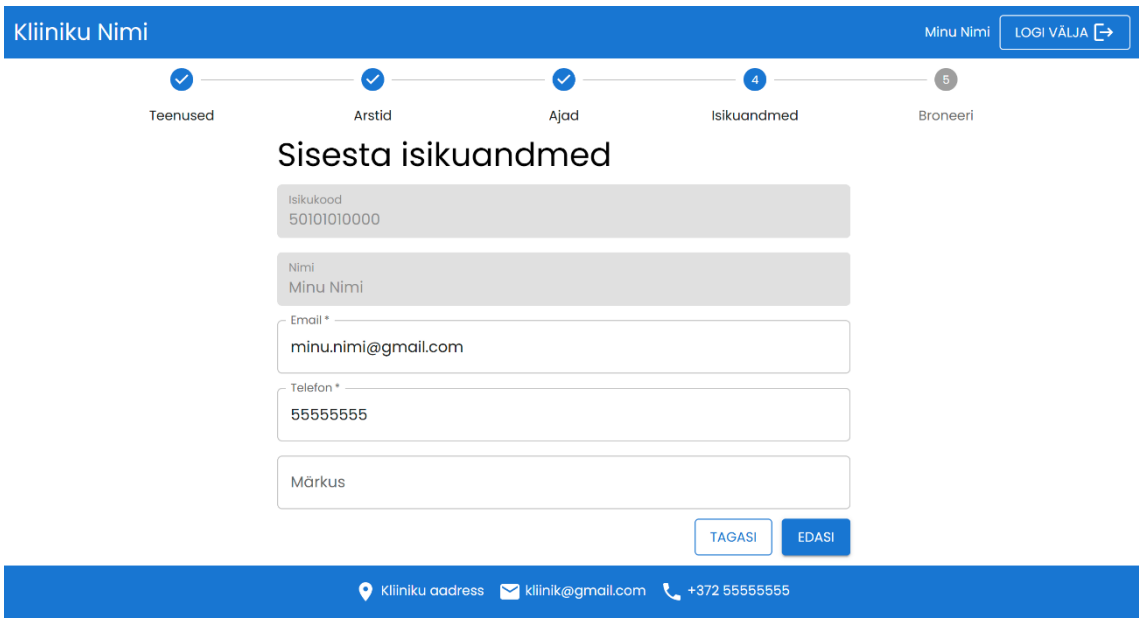
Joonis 14. Broneerimissüsteemi broneeritava aja valimise vaade.

Aegade valiku vaade võimaldab filtreerida aegu nii päevade kui ka ajavahemiku alusel. Esialgselt on kalendris valitud esimest aega sisaldav päev. Kalendri puhul on otsustatud kuvada vabu aegu sisaldavaid päevi halli taustaga. Tausta värv on hoitud sarnane nii

teenuse kui ka arstide valikute taustaga, et kasutajal saaks automaatselt tekkida arusaam, et just need päevad kalendris on valitavad.

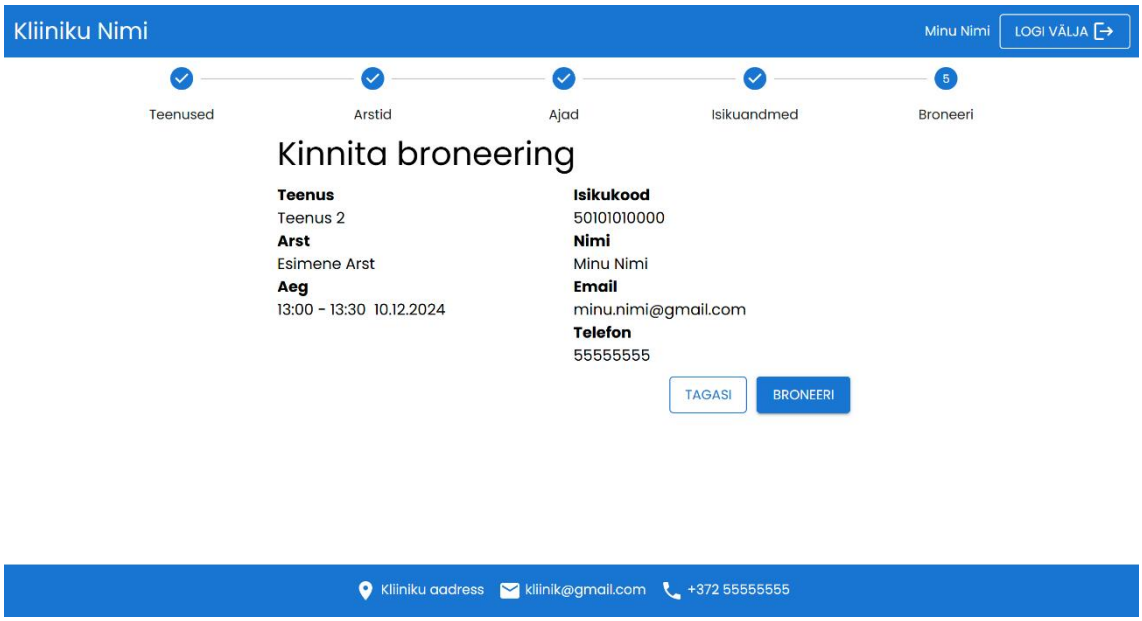
Ajavahemiku valimise sisend moodustatakse iga valitud päeva puhul automaatselt, arvestades sellel päeval saadaolevaid vabu aegu. Teisisõnu moodustatakse ajavahemiku valitavad sisendväärtused vastavalt valitud päeva vabadele visiidi algusaegadele.

Olles valinud sobiva aja, liigutakse edasi isikuandmete sisestamise vaatesse (Joonis 15).



Joonis 15. Broneerimissüsteemi isikuandmete sisestamise vaade.

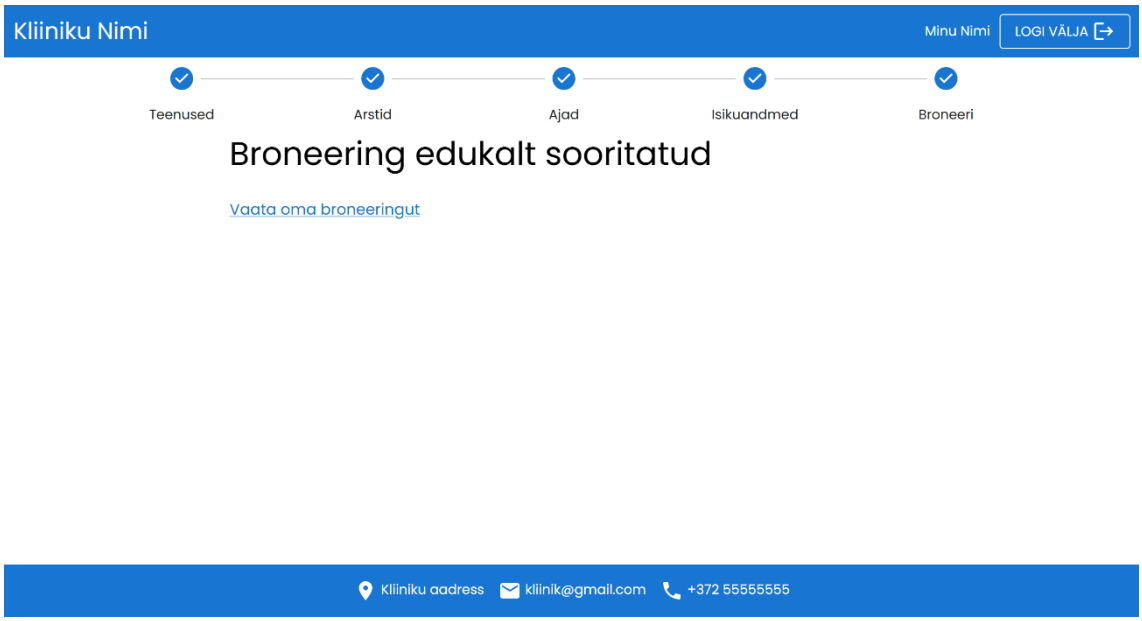
Isikuandmete sisestamisel täidetakse isikukood ja nimi automaatselt autentimise andmetega. Kasutajal on kohustuslik sisestada e-mail ja telefon, märkuse lisamine on valikuline. Olles sisestanud kõik kohustuslikud väljad, on kasutajal võimalik liikuda edasi ning näha oma broneeringu ülevaadet (Joonis 16).



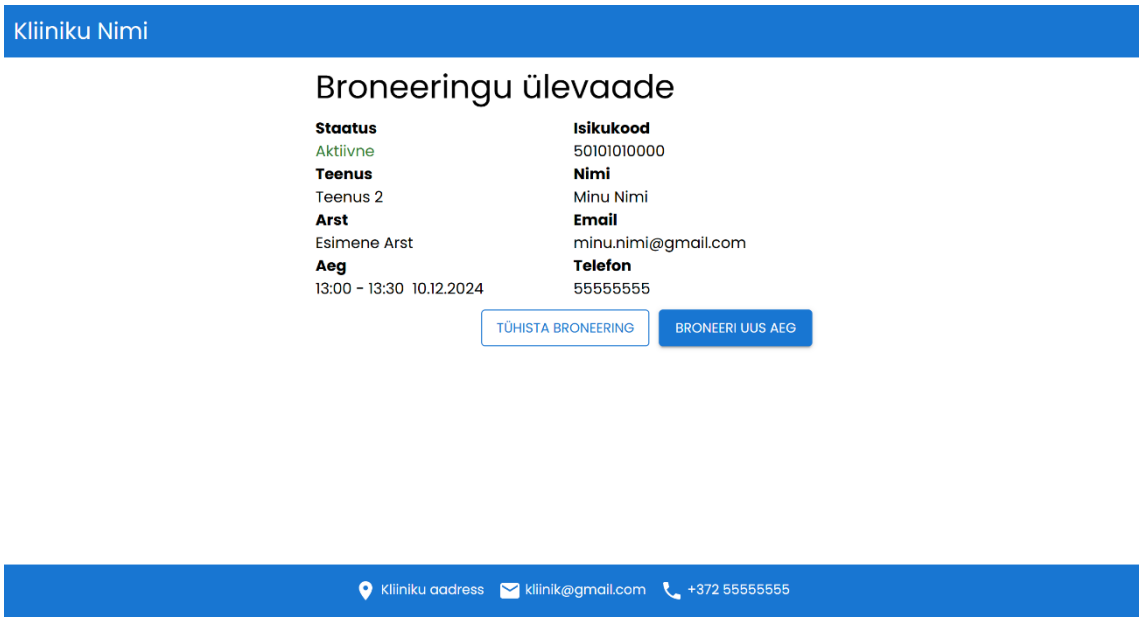
Joonis 16. Broneerimissüsteemi broneeringu kinnitamise vaade.

Broneeringu kinnitamise vaade on oluline, kuna hetkel broneerimissüsteem erinevate sammude vaadetes eelneva sammu käigus tehtud otsuseid ei kuva. Näiteks aja valiku vaates puudub kontekst, millisele teenusele aega broneeritakse. Sellist käitumist võib kindlasti lugeda broneeringusüsteemi disaini suurimaks nõrkuseks, kuid prototüübi arendusel otsustati ajaliste piirangute tõttu nende disainiotsustega mitte tegeleda. Kogu broneeringuga seonduvat infot on võimalik korraga näha vaid vahetult enne broneeringut.

Eduka broneeringu korral kuvatakse kasutajale vastav teade (Joonis 17). Vaade võimaldab ka navigeerimist broneeringu ülevaatesse (Joonis 18).



Joonis 17. Broneeringusüsteemi eduka broneeringu sooritamise vaade.



Joonis 18. Broneerimissüsteemi broneeringu ülevaate vaade.

Broneeringu ülevaade võimaldab kliendil olla kursis temale oluliste broneeringu detailidega.

4.2.3 Esirakenduse vaated mobiilseadmetes

Loodud vaadete puhul on oluline nende väljanägemine ka mobiilseadmetes. Erinevalt arvutiekraanist on telefoni ekraanid kitsamad ning vajavad seetõttu elementide optimaalselt paigutamisel tihtipeale teistsugust lähenemist. Järgnevalt on kirjeldatud mobiilivaadete kujundamise tehnilisemat poolt.

Mobiilivaate all mõeldakse prototüübi raames brauseriakna vaadet, mille DIP (*device-independent pixels*) laius on 600 pikslit või vähem. DIP ehk seadmetest sõltumatud pikslid on abstraktne mõõtühik, mida kasutatakse veebiarenduses ekraanide pikslitiheduse erinevuste ühtlustamiseks [29].

Kitsamate ekraanide puhul kujundatakse vajadusel elementide stiili tavapärasest erinevalt läbi CSS klassi reeglite [30] (Joonis 19).

```
@media (max-width: 600px) {
    .service-header {
        justify-content: space-between;
    }
}
```

Joonis 19. Ekraanilaiusel põhineva stiilireegli kasutamise näide.

Lisaks elementide stiilile soovitakse sõltuvalt ekraanilaiusest kohandada ka nende struktuuri. Selleks on vajalik võimaldada React komponentidele ligipääs brauseriakna mõõtmetele. Prototüübi raames luuakse selle realiseerimiseks WindowWidthContext (Joonis 20), mille väärtusi WindowWidthProvider (Joonis 21) ekraanimõõtmete muutuste korral uuendab.

```
import { createContext } from 'react';

type WindowWidthContextType = {
    windowWidth: number;
    isMobileView: boolean;
};

export const WindowWidthContext = createContext<WindowWidthContextType>({
    windowWidth: 0,
    isMobileView: false
});
```

Joonis 20. WindowWidthContext.ts fail.

```tsx
import React, {useEffect, useState} from 'react';
import {WindowWidthContext} from './WindowWidthContext.ts';

export default function WindowWidthProvider({children}: {children: React.ReactNode}) {
    const [windowWidth, setWindowWidth] = useState(window.innerWidth);
    const [isMobileView, setIsMobileView] = useState(window.innerWidth <= 600);

    useEffect(() => {
        const handleResize = () => {
            setWindowWidth(window.innerWidth);
            setIsMobileView(window.innerWidth <= 600);
        };

        window.addEventListener('resize', handleResize);

        return () => window.removeEventListener('resize', handleResize);
    }, []);

    return (
        <WindowWidthContext.Provider value={{ windowWidth, isMobileView }}>
            {children}
        </WindowWidthContext.Provider>
    );
}
```

Joonis 21. WindowWidthProvider.tsx fail.

Broneerimissüsteemi mobiiliseadmete vaated on välja toodud lisas (Lisa 3).

5 Hinnang loodud rakendusele

Iga rakenduse arendusel on eesmärgiks saada valmis võimalikult hea rakendus. Ideaalis vastaks loodud rakendus kõikidele nõuetele, kõik komponendid toimiksid ideaalselt, rakendus arvestaks kõikide võimalike olukordadega ning käsitleks neid perfektselt. Rakendus oleks lõplikult kaetud testidega ning võimaldaks endas teha mis iganes muutuseid mugavaimal viisil. Reaalsuses tulevad aga arenduse protsessis ette tihtipeale ootamatused ning ajalised piirangud, mis sunnivad tegema arenduse faasis kompromisse. Järgnevalt analüüsitakse lõputöö raames loodud veebirakendust.

5.1 Nõuetele vastavus

Lõputöö raames loodud prototüüp vastab kõigile analüüsi faasis käsitletud nõuetele. Rakendusele esitatud nõuded aitasid hoida fookust olulistel teemadel ning tegid seeläbi võimalikuks jõuda lõputöö käigus prototüübi arendusega valmis.

5.2 Nõrkused

Peaaegu kõiki tarkvaralisi lahendusi on võimalik arendada lõputult. Ükski rakendus ei ole kunagi valmis. Alati on võimalik leida aspekte, mida oleks võimalik lahendada paremini ja funktsionaalsust, mida oleks võimalik lisada juurde.

Isegi kui arendatavasse prototüüpi pole ajaliste piirangute tõttu enam võimalik sisse tuua parandusi, on sellegipoolest oluline teha endale süsteemi nõrkused selgeks ning olla nendest teadlik. Nõrkuste analüüs aitab kinnistada arusaama prototüübi toimimisest ning on oluline osa programmeerija arengust. Järgnevalt analüüsitakse broneeringusüsteemi peamisi nõrku kohti.

Tagarakenduse poole pealt vajaks täiendust vigade töötlemine. Hetkel viskab tagarakendus vigade korral erindi (*exception*), mis kontrollerile lisatud erindi filtri [31] poolt kinni püütakse ning *Bad Request* [32] vastuseks ümber teisendatakse. Vastusesse antakse ka kaasa vea sõnum, mis on konfigureeritud olema alati „Päring ebaõnnestus“. Erandina käsitletakse teisiti broneeringuliidese poolt tagastatud vigu, mille sõnumid

säilitatakse ning päringu vastusesse tavapärase sõnumi asemel lisatakse. Ideaalis võiks aga vea sõnumid alati kirjeldada täpsemalt ebaõnnestumise põhjust. Vastuse HTTP staatuse kood [32] võiks samuti olla sõltuv vea iseloomust. Hetkel on see aga alati 400.

Esirakenduse poolelt tuleks parandada tegelemist olukordadega, mil tagarakenduse kaudu küsitud nimekiri andmetest on tühi. Nii teenuste, arstide kui ka broneeritavate aegade vaated tegelevad andmete küsimisega läbi tagarakenduse. Kui näiteks tagarakendus tagastab teenuste vaates tühja teenuste nimekirja, ei käsitleta seda olukorda esirakenduse poolelt kuidagi teisiti. Ühtegi teenust kasutajale pole võimalik kuvada ning kuna päring oma olemuselt õnnestus, ei kuvata ka ühtegi veateadet. Selliste olukordadega tuleks kindlasti tegeleda ja edaspidise arenduse käigus nendega arvestada.

5.3 Edasiarenduse võimalused

Ühe kõige olulisema edasiarendusena vajaks loodud broneeringusüsteem kindlasti enne Kohila Meedik OÜ hambaravikeskusega sidumist testidega kaetust. Prototüüpi arenduse faasis, läbi esirakenduse manuaalselt rakendust testides, ei ole vigadega kokkupuude tülikas protsess. Veaga on võimalik koheselt tegeleda ning see likvideerida. *Live*-keskkonna puhul on aga vigade avastamine palju tülikam. Isegi kui kliendi poolt avastatud viga on oma olemuselt väike ning on võimalik vähese vaevaga ja kiirelt ära parandada, jääb kliendile esmamulje süsteemist ikkagi negatiivne.

Iga arendaja teeb arenduse faasis vigu ning oleks naiivne eeldada, et arenduse faasis suudetakse testide olemasoluta üles leida kõik süsteemis esinevad vead. Lisaks pakub testidega kaetus suuremat kindlust arendajale ka muutuste tegemisel. Mingisuguse lisafunktsionaalsuse lisamisel ei pea enam testima kõiki varasemaid funktsionaalsusi, vaid on seda protsessi võimalik automatiseerida. Lõputöö raames otsustati ajaliste piirangute tõttu teste mitte koostada, kuid rakenduse testidega kaetus nii ühik- kui ka integratsioonitestide tasemel peaks olema kindlasti suur prioriteet enne rakenduse reaalset kasutuselevõttu.

Broneeringusüsteemi funktsionaalsuse poolelt oleks võimalik pakkuda kasutajatele võimalust näha oma broneeringute ajalugu. Broneeringute ajaloo nägemine võimaldaks kasutajatel saada kiire ülevaade oma varasematest broneeringutest. Samuti võimaldaks see varasemate broneeringute põhjal lihtsustada kordusvisiitide broneerimise protsessi.

5.4 Rakenduse kasutuselevõtu majanduslikud kulud

Järgnevalt analüüsitakse veebirakenduse kasutusevõtu kulusid, lähtudes eeldatavast broneeringusüsteemi kasutatavast klientide arvust kuus.

5.4.1 Dentase broneeringuliidese kulu

Dentase poolt pakutava broneeringuliidese kasutamine vastu päris kliiniku andmeid on tasuline protsess. Broneeringuliidese kasutamise tasu on makseteenust mitte-kasutavate broneeringute puhul 0,45€ iga tehtud broneeringu kohta.

Kohila Meedik OÜ tegeleb kuus keskmiselt 600 visiidiga.

Majanduslike kulude arvutamisel on lõputöö raames eeldatud, et rakenduse kasutuselevõtt suunaks broneeringusüsteemi kasutama kolmandiku klientidest. Kolmandiku klientide puhul oleks võimalik eeldada, et kuus tehakse keskmiselt broneeringusüsteemi kasutades 200 broneeringut.

Seega oleks eeldatud tingimustel broneeringusüsteemi kasutuselevõtu hind broneeringuliidese koha pealt keskmiselt umbes 90€ kuus.

5.4.2 Autentimisteenuste kulu

Lõputöö raames otsustati teostada autentimist Smart-ID teenuse kaudu. Kuna pikemas perspektiivis on aga oluline pakkuda klientidele autentimisvõimekust ka muudel viisidel, lähtutakse kulu arvutamisel ka prototüübi raamest välja jäänud autentimisteenustest.

Teenuste maksumuse arvutamisel on lähtutud SK ID Solutions pakutavatest hindadest [33]. SK ID Solutions on eesti ettevõte, mis keskendub turvaliste digitaalsete identiteedilahenduste pakkumisele.

Nii Smart-ID kui Mobiil-ID puhul piisaks SK ID Solutions poolt pakutavast odavaimast paketist. Smart-ID odavaim pakett maksab 55€ kuus ning võimaldab teha kuni 550 autentimistehingut. Mobiil-ID odavaim pakett maksab 30€ kuus ning võimaldab teha 300 tehingut. Mõlemad paketid katavad ära eeldatud klientide arvu kuus.

ID-kaardiga autentimisel ei ole vaja kasutada SK ID Solutions vahendusteenuseid, seega sellega eraldi lisakulusid ei kaasne.

Autentimisteenuste kulu on eeldatud klientide arvu juures kokku arvutatult 85€ kuus.

5.4.3 Veebirakenduse hoiustamise kulu

Et kasutajatel oleks võimalik rakendust kasutada, peab see olema kuskil hoiustatud. Kuna lõputöö raames ei analüüsita süsteemi reaalset kasutuselevõttu ning vaadeldakse prototüüpi kui arenduse faasis olevat teenust, ignoreeritakse kulu arvutamisel tulevikus lisanduvat hoiustamise hinda.

5.4.4 Kulude kokkuvõte

Kulude kokkuvõttes käsitletakse erinevate tasuliste komponentide maksumust kokkuarvutatult.

Broneeringusüsteemi kasutuselevõtt läheks eeldatud tingimustel ettevõttele Kohila Meedik OÜ maksma keskmiselt 175€ kuus.

6 Kokkuvõte

Käesoleva bakalaureusetöö tulemusena loodi hambaravikeskusele Kohila Meedik OÜ broneerimissüsteemi prototüüp. Prototüübi arendus jäi lõputöö raames vastu testkeskkonda, hambaravikeskuse kasutuselevõtu samme kirjeldati vaid teoreetiliselt.

Lõputöö käigus selgitati hetkeolukorda ja selle miinuseid. Sõnastati prototüübi nõuded ning kirjeldati protsessi parima lahenduseni jõudmiseks. Analüüsi käigus kaaluti erinevaid lahendusi ning tehti nende seast põhjendatud valik. Analüüsile järgnes prototüübi arendus, mille tulemusena valmis nii esi- kui ka tagarakendus. Arenduse käigus tehtud otsuseid selgitati põhjendatult. Viimaks hinnati loodud prototüübi nõuetele vastavust, toodi välja selle nõrkused ja edasiarenduse võimalused ning arvutati broneeringusüsteemi kasutuselevõtu majanduslikud kulud.

Lõputöö seatud eesmärk täideti ning prototüüp on võimeline täitma kõiki sellele esitatud nõudeid. Prototüüp on võimeline korraldama broneeringute tegemist, lihtsustades seeläbi nii kliendi kui ka hambaravikeskuse tööd.

Kasutatud kirjandus

[1] Connected OÜ, „Dentas Site“ [Võrgumaterjal]. Saadaval: https://connected.ee/dentas-site. [Kasutatud 22.09.2024].

[2] Kohila Meedik OÜ, „Kohila hambakliinik“ [Võrgumaterjal]. Saadaval: https://kohilahambakliinik.ee. [Kasutatud 22.09.2024].

[3] Connected OÜ, „Töötamine online broneeringutega“ [Võrgumaterjal]. Saadaval: https://www.connected.ee/Online%20broneeringud.pdf. [Kasutatud 22.09.2024].

[4] SK ID Solutions, „Smart-ID integreerimise protsess“ [Võrgumaterjal]. Saadaval: https://www.smart-id.com/et/e-teenuste-pakkujale/integreerimisprotsess. [Kasutatud 09.10.2024].

[5] SK ID Solutions, „Smart-ID client libraries“ [Võrgumaterjal]. Saadaval: https://github.com/SK-EID/smart-id-documentation/wiki/Client-libraries. [Kasutatud 09.10.2024].

[6] D. Bogatykh, „smart-id-net-client“ [Võrgumaterjal]. Saadaval: https://github.com/bogatykh/smart-id-net-client. [Kasutatud 09.10.2024].

[7] React, „React“ [Võrgumaterjal]. Saadaval: https://react.dev. [Kasutatud 11.10.2024].

[8] Ant Design, „Ant Design“ [Võrgumaterjal]. Saadaval: https://ant.design. [Kasutatud 11.10.2024].

[9] Chakra, „Chakra UI“ [Võrgumaterjal]. Saadaval: https://www.chakra-ui.com. [Kasutatud 11.10.2024].

[10] MUI, „MUI“ [Võrgumaterjal]. Saadaval: https://mui.com. [Kasutatud 11.10.2024].

[11] Google, „Material Design“ [Võrgumaterjal]. Saadaval: https://m3.material.io. [Kasutatud 11.10.2024].

[12] Microsoft, „.NET platform“ [Võrgumaterjal]. Saadaval: https://dotnet.microsoft.com/en-us. [Kasutatud 15.10.2024].

[13] GitHub, „GitHub“ [Võrgumaterjal]. Saadaval: https://github.com. [Kasutatud 04.10.2024].

[14] GitLab, „GitLab“ [Võrgumaterjal]. Saadaval: https://about.gitlab.com. [Kasutatud 04.10.2024].

[15] Atlassian, „BitBucket“ [Võrgumaterjal]. Saadaval: https://bitbucket.org. [Kasutatud 04.10.2024].

[16] World Wide Web Consortium, „Cascading Style Sheets“ [Võrgumaterjal]. Saadaval: https://www.w3.org/TR/CSS/#css. [Kasutatud 07.10.2024].

[17] W3Schools, „What is JSON?“ [Võrgumaterjal]. Saadaval: https://www.w3schools.com/whatis/whatis_json.asp. [Kasutatud 07.10.2024].

[18] Library of Congress, „ISO 639-1: Codes for the Representation of Names of Languages“ [Võrgumaterjal]. Saadaval: https://id.loc.gov/vocabulary/iso639-1.html. [Kasutatud 03.11.2024].

[19] Microsoft, „Caching in .NET“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/dotnet/core/extensions/caching. [Kasutatud 15.10.2024].

[20] S. Chacon, „Git Tools - Submodules“ [Võrgumaterjal]. Saadaval: https://git-scm.com/book/en/v2/Git-Tools-Submodules. [Kasutatud 09.10.2024].

[21] Microsoft, „An introduction to NuGet“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/nuget/what-is-nuget. [Kasutatud 09.10.2024].

[22] SK ID Solutions, „Environment technical parameters“ [Võrgumaterjal]. Saadaval: https://github.com/SK-EID/smart-id-documentation/wiki/Environment-technical-parameters. [Kasutatud 09.10.2024].

[23] Microsoft, „Session and state management in ASP.NET Core“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-8.0. [Kasutatud 15.10.2024].

[24] TypeScript, „TypeScript“ [Võrgumaterjal]. Saadaval: https://www.typescriptlang.org. [Kasutatud 10.10.2024].

[25] Vite, „Vite“ [Võrgumaterjal]. Saadaval: https://vite.dev. [Kasutatud 10.10.2024].

[26] Axios, „Axios Documentation“ [Võrgumaterjal]. Saadaval: https://axios-http.com/docs/intro. [Kasutatud 10.10.2024].

[27] Mozilla Developer Network, „HTTP“ [Võrgumaterjal]. Saadaval: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Kasutatud 05.10.2024].

[28] GeeksForGeeks, „How to Create Mail and Phone Link in HTML“ [Võrgumaterjal]. Saadaval: https://www.geeksforgeeks.org/how-to-create-mail-and-phone-link-in-html. [Kasutatud 14.11.2024].

[30] W3Schools, „CSS Media Queries“ [Võrgumaterjal]. Saadaval: https://www.w3schools.com/css/css3_mediaqueries.asp. [Kasutatud 07.10.2024].

[31] Microsoft, „Filters in ASP.NET Core - Exception filters“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-8.0#exception-filters. [Kasutatud 15.10.2024].

[32] Mozilla Developer Network, „HTTP Status Codes“ [Võrgumaterjal]. Saadaval: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. [Kasutatud 05.10.2024].

[33] SK ID Solutions, „Teenused - Autentimine ja allkirjastamine“ [Võrgumaterjal]. Saadaval: https://portal.skidsolutions.eu/order/authentication-signing. [Kasutatud 09.10.2024].

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

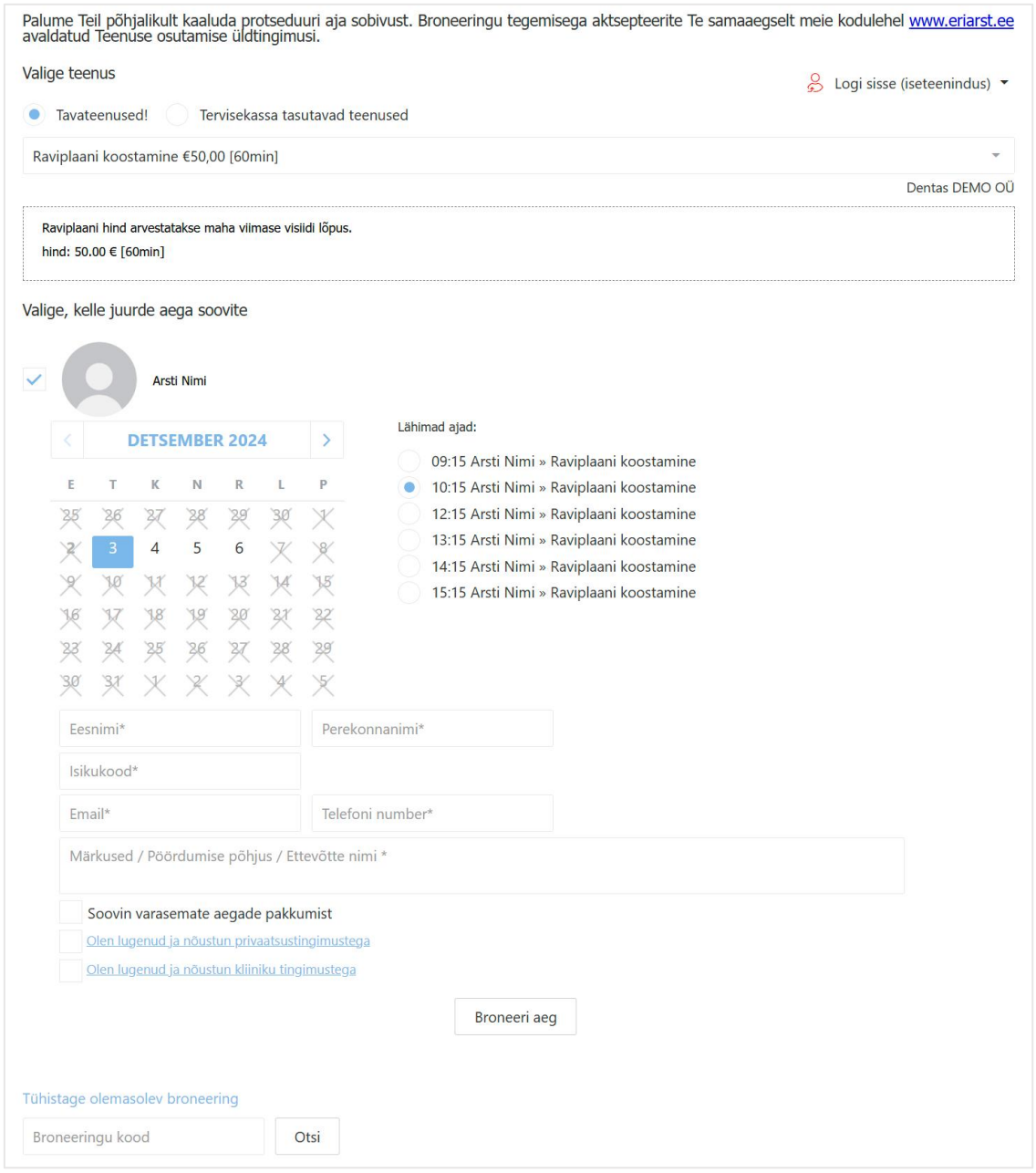
Mina, Oskar Laak

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kohila Meedik OÜ broneerimissüsteemi prototüüp“, mille juhendaja on Kersti 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.

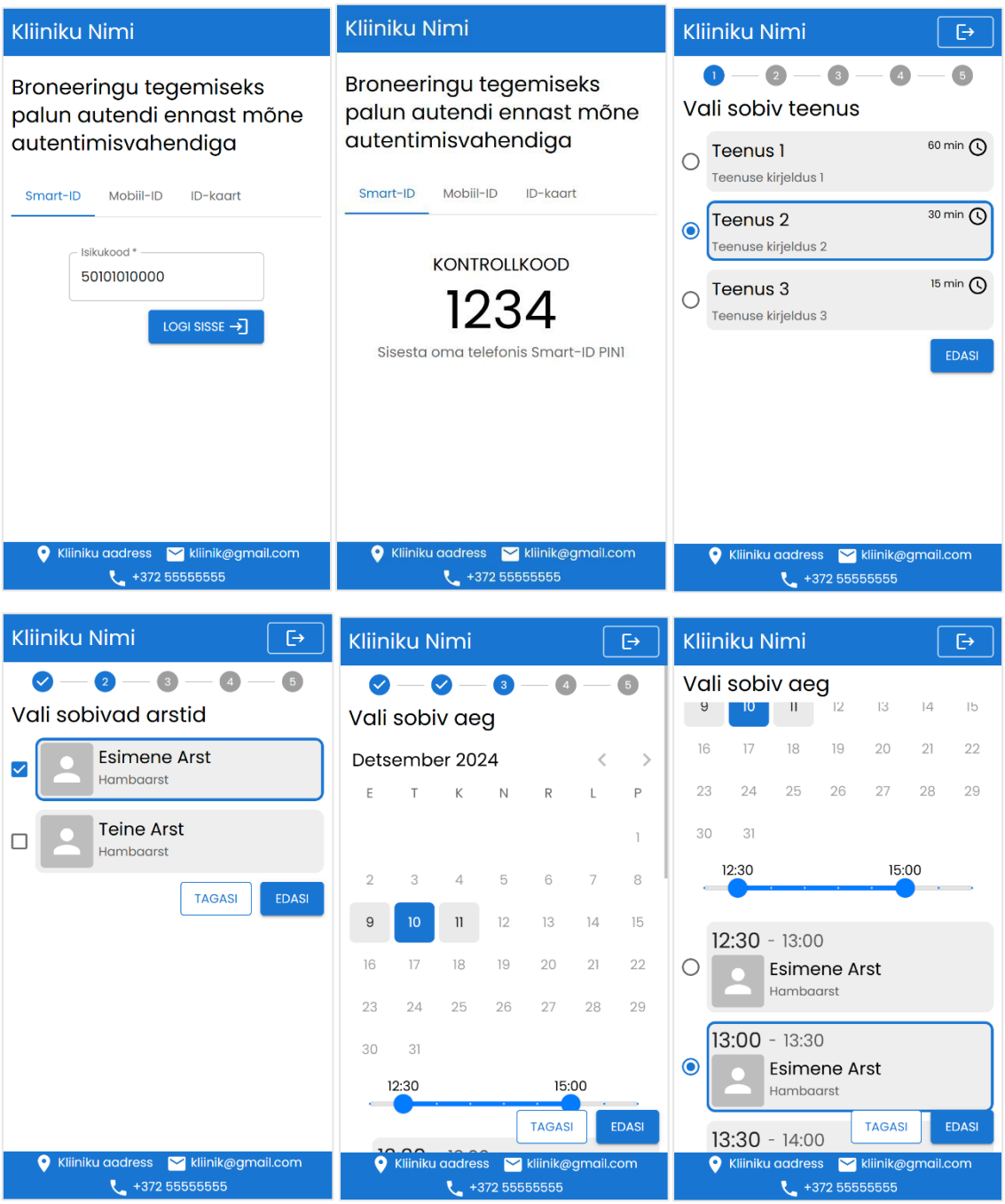
06.01.2025

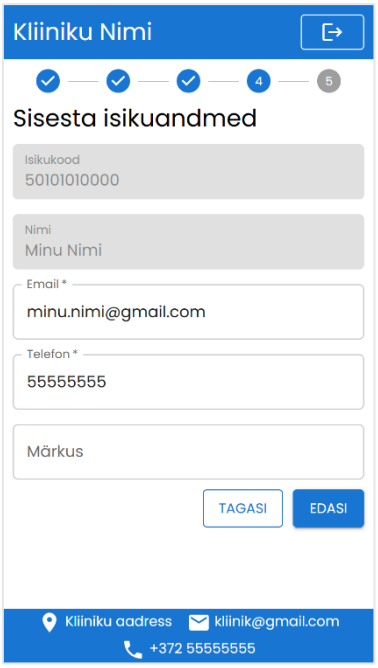
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või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

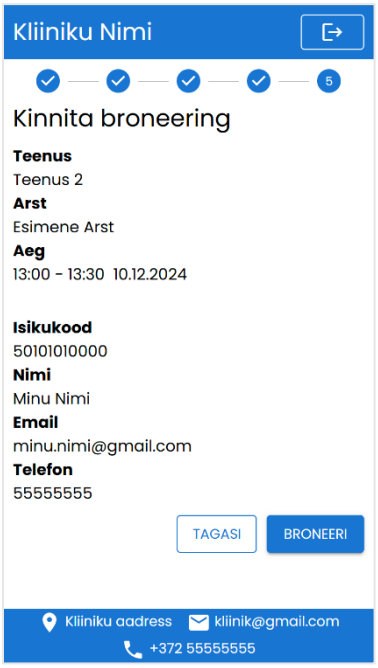
Lisa 2 – Dentase veebirakenduse vaade

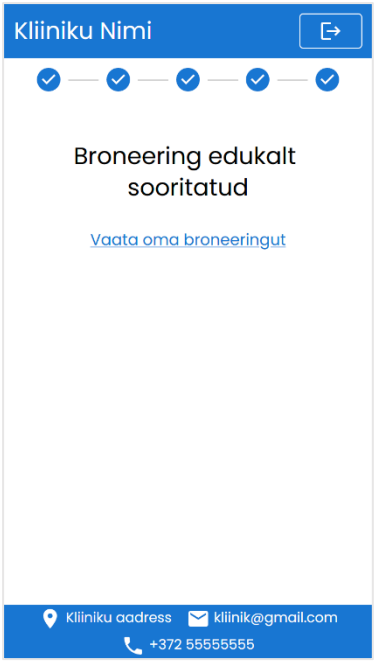


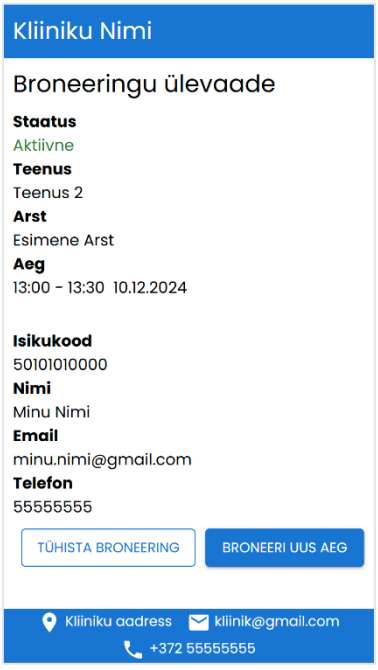
Lisa 3 – Loodud veebirakenduse mobiilivaated




Kliiniku Nimi
4
5
Sisesta isikuandmed
Isikukood
50101010000
Nimi
Minu Nimi
Email *
minu.nimi@gmail.com
Telefon *
55555555
Märkus
TAGASI
EDASI
Kliiniku aadress
kliinik@gmail.com
+372 55555555


Kliiniku Nimi
5
Kinnita broneering
Teenus
Teenus 2
Arst
Esimene Arst
Aeg
13:00 - 13:30 10.12.2024
Isikukood
50101010000
Nimi
Minu Nimi
Email
minu.nimi@gmail.com
Telefon
55555555
TAGASI
BRONEERI
Kliiniku aadress
kliinik@gmail.com
+372 55555555


Kliiniku Nimi
Broneering edukalt sooritatud
Vaata oma broneeringut
Kliiniku aadress
kliinik@gmail.com
+372 55555555


Kliiniku Nimi
Broneeringu ülevaade
Staatus
Aktiivne
Teenus
Teenus 2
Arst
Esimene Arst
Aeg
13:00 - 13:30 10.12.2024
Isikukood
50101010000
Nimi
Minu Nimi
Email
minu.nimi@gmail.com
Telefon
55555555
TÜHISTA BRONEERING
BRONEERI UUS AEG
Kliiniku aadress
kliinik@gmail.com
+372 55555555