TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihhail Kuznetsov 193911IADB

Veebirakenduse arendamine aja broneerimiseks juuksurisalongis

Bakalaureusetöö

Juhendaja: Einar Kivisalu
MSc

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: Mihhail Kuznetsov

15.05.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua veebirakendus juuksurisalongile, mis võimaldab klientidel mugavalt veebis aega broneerida.

Esimeses osas kirjeldab autor põhjalikult töös käsitletavat probleemi, projekti eesmärki ja metoodikat probleemi lahendamiseks. Teises osas kirjeldab ja analüüsib autor hetkel olemasolevaid veebirakendusi, mis lahendavad sama ülesannet kui loodav veebirakendus. Kolmandas osas kirjeldab autor tehnoloogiaid, mida valiti eesmärkide saavutamiseks ja nende valiku põhjuseid. Neljandas osas kirjeldab autor lahenduse elluviimist, sealhulgas arhitektuuri, andmebaasi, tagaplaani ja esipaneeli arendamist. Viimases osas vaatab autor üle kasutajate testimise tulemused ja esile toob peamised ideed, mis võivad tulevikus rakenduse funktsionaalsusesse integreeritud saada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 13 joonist.

Abstract

Development of a Web Application for Booking Time in a Barbershop

The aim of this bachelor's thesis is to develop an online application for a barbershop that allows customers to conveniently schedule appointments online.

In the first part, the author describes in detail the problem discussed in the dissertation, outlines the project's objectives and methodology used to solve the problem. In the second part, the author describes and analyzes alternative web applications currently available that solve the same problem as the web application being developed. In the third part, the author describes the technologies selected to achieve the goals and the reasons for the choice. In the fourth part, author describes the implementation of the solution, including the development of architecture, database, backend, and frontend. In the fifth section, the author examines the results of user testing and identifies key ideas that could be implemented in the application's functionality in the future.

The thesis is in Estonian and contains 33 pages of text, 6 chapters, 13 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*. Rakendusliides.
AJAX	*Asynchronous Javascript and XML*. Asünkroonne Javascript ja XML.
ASP.NET Core	Raamistik veebirakenduste kirjutamiseks.
BLL	*Business Logic Layer,* Äriloogika kiht
CSS	*Cascading Style Sheets*. Märgistuskeeles kirjutatud HTML dokumendikuvamise stiili kirjeldav keel.
DAL	*Data Access Layer,* Andmetele kättesaadavuse kiht
HTTP	*Hyper Text Transfer Protocol*. Protokoll teabe edastamiseks arvutivõrkudes
HTML	*Hypertext Markup Language*. Veebilehtede märgistuskeele standard.
JSON	*JavaScript Object Notation*. formaat andmete salvestamiseks ja transportimiseks.
LINQ	*Language-Integrated Query*, päringukeel
MVC	*Model-view-controller*, mudel-vaade-kontroller disainimuster.
MVCC	*Multiversion Concurrency Control*, see on samaaegsuse kontrolli meetod, mida tavaliselt kasutavad andmebaasihaldussüsteemid, et pakkuda samaaegset juurdepääsu andmebaasile
MVP	*Minimum Viable Product*. Toode, millel on piisavalt funktsioone, et meelitada ligi varakult kasutusele võtnud kliente ja kinnitada tooteidee tootearendustsükli alguses.
.NET	Microsofti poolt pakutav tarkvaraarendusplatvorm.
ORM	*Object-Relational Mapping,* Programmeerimistehnika, mille puhul kasutatakse metaandmete deskriptorit objektikoodi ühendamiseks relatsiooniandmebaasiga
ORDBMS	*Object-Relational database management system*, Programm, mida kasutatakse relatsiooniandmebaaside loomiseks, värskendamiseks ja haldamiseks.

REST	*Representational state transfer*, tarkvara arhitektuurne stiil.
SQL	Struktuurpäringukeel.
WAL	*Web Application Layer,* Kiht, mis sisaldab veebilehti, veebiteenuseid ja muid kasutajaga suhtlemiseks vajalikke komponente

Sisukord

Jooniste loetelu

1 Sissejuhatus

Viimastel aastatel on ilutööstus kiiresti arenenud ja paljud inimesed pöörduvad juukselõikuse saamiseks professionaalide poole. Kasvava nõudluse tõttu seisavad juuksurid silmitsi väljakutsega tõhusalt broneeringuid hallata, eriti tippperioodidel. Traditsioonilised kohtumise kokkuleppimise meetodid, nagu telefonikõned ja kohtumised, on osutunud ebamugavaks nii ettevõtte omanikule kui ka klientidele[1].

Kõik broneerimissüsteemid ei ole aga ühesugused ja ettevõtetel võib olla keeruline leida oma konkreetsetele vajadustele vastavat lahendust. Sellest tulenevalt tekib vajadus veebirakenduse järele, mis võimaldab klientidel veebis aegu broneerida, lihtsustades protsessi ja säästes aega. Antud lõputöö lähtetingimused on sellised, et tellija esitas tehnilise ülesande, milles kirjeldati kõiki nõudeid, mida loodud veebirakenduses tuleb realiseerida.

Rääkides täpsemalt lõputöö esimesest osast, peatub autor lõputöös käsitletud probleemil. Probleem seisneb juuksurisalongi praeguse broneerimisprotsessi ebaefektiivsuses, kus kliendid peavad helistama või füüsiliselt salongi tulema, et aega kokku leppida.

Probleemi lahendamiseks kasutatud metoodika osas toob autor välja eesmärgi saavutamiseks tehtud sammud. Nende hulka kuuluvad olemasolevate lahenduste analüüs, allikate uurimine ning minimaalse elujõulise toote väljatöötamine, lähtudes rakendusest, klientide soovidest ja analüüsi tulemustest. Samuti rõhutab autor saadud tagasiside põhjal testimise ja täiustamise olulisust.

Lõputöö teises osas analüüsib autor hetkel saadaolevaid alternatiivseid veebirakendusi, mis lahendavad arendatavaga sama probleemi. See analüüs hõlmab rakenduste funktsioonide, funktsionaalsuse ja kasutuskogemuse võrdlemist, keskendudes valdkondade väljaselgitamisele, kus arendatavat veebirakendust saab täiustada.

Liikudes edasi lõputöö kolmanda osa juurde, kirjeldab autor eesmärkide saavutamiseks valitud tehnoloogiaid ja valiku põhjuseid. See hõlmab otsust kasutada *C#* programmeerimiskeelt sisemise rakenduste arendamiseks ja *ASP.NET Core*'i esiotsa jaoks.

Autor kirjeldab nende tehnoloogiate eeliseid projekti jaoks.

Lõputöö neljandas osas kirjeldab autor lahenduse teostust. See hõlmab arhitektuuri, andmebaasi, taustaosa ja kasutajaliidese arendamist. Viiendas osas vaatab autor läbi kasutajatestimise tulemused ning toob välja võtmeideed, mida saab tulevikus rakenduse funktsionaalsuses rakendada.

1.1 Taust

Tarkvaraarendajana pöördus minu poole juuksuritöökoja omanik, kes soovis luua klientidele toimiva veebirakenduse, et broneerida kohtumisi veebis. Antud lõputöö probleem seisneb selles, et klient vajab mugavat ja toimivat rakendust juuksurisse aja kokkuleppimiseks. Käesoleva töö eesmärgiks on analüüsida olemasolevaid süsteeme, kujundada ja arendada veebirakendust, mis oleks klientidele mugav kasutada ja vastaks kliendi nõudmistele.

Olemasolevate rakenduste ülevaatuse põhjal seavad klient ja autor enne veebirakenduse väljatöötamist rakendusele järgmised nõuded:

- Veebirakendus peab võimaldama klientidel näha saadaolevaid juuksurikohtumiste aegu.
- Veebirakendus peab võimaldama klientidel valida koosoleku kuupäeva ja kellaaja.
- Veebirakendus peab võimaldama klientidel broneeringuid tühistada.
- Veebirakendus peab võimaldama klientidel luua konto ja salvestada oma broneeringute ajalugu.
- Veebirakendus peab võimaldama juuksuripoe omanikul või nende määratud administraatoritel kohtumisi hallata, sealhulgas ajavahemikke lisada ja eemaldada.
- Veebirakendusel peab olema kasutajasõbralik liides nii klientidele kui ka juuksuritele.
- Veebirakendus peab olema turvaline ja kaitsma kliendi andmeid.

1.2 Probleemi tutvustus

Selle lõputöö probleemipüstituseks on vajadus funktsionaalse veebirakenduse järele, mis hõlbustaks juuksurisalongi klientide online-aja broneerimist. Praegune aeg telefonikõnede või isiklike külastuste kaudu broneerimise protsess võib olla aeganõudev ja ebamugav nii salongi töötajatele kui ka klientidele.

Juuksurit omav klient soovib parandada oma klientide broneerimisprotsessi ja hõlbustada salongi soenguaegade planeerimist. Seetõttu on selle probleemi lahendamiseks vaja välja töötada veebirakendus, mis võimaldab klientidel hõlpsalt veebis kohtumisi ajastada. Lõputöö eesmärk on uurida sellise veebirakenduse arendamist ja pakkuda terviklikku lahendust, mis võib parandada broneerimisprotsessi nii salongi personali kui ka klientide jaoks.

1.3 Metoodika

Veebirakenduse arendamiseks oli vaja teha erinevaid analüüse ja uuringuid. Esiteks viidi läbi olemasolevate lahenduste analüüs, et tuvastada turul esinevad lüngad ja teha kindlaks peamised omadused, mis veebirakendusel olema peaksid.

Teine analüüs, mis viidi läbi, oli tehniline analüüs, et selgitada välja kõige sobivamad tehnoloogiad veebirakenduste arendamiseks. Antud analüüsis võeti arvesse selliseid tegureid nagu turvalisus ja juurutamise lihtsus tagamaks, et valitud tehnoloogiad suudavad toetada kvaliteetse ja kliendi vajadustele vastava veebirakenduse väljatöötamist.

Teine oluline samm arendusprotsessis oli erinevatest allikatest pärinevate uuringute läbiviimine teemakohase teabe kogumiseks. See uuring andis ülevaate veebirakenduste arendamise viimastest trendidest ja parimatest praktikatest, mis oli oluline arendusprotsessi käigus tehtud tehniliste otsuste informeerimisel.

Võttes arvesse projekti mastaapi, analüüsi tulemusi ja tellija nõudmisi, koostati veebirakenduse kirjeldus. See kirjeldus tõi välja veebirakenduse põhifunktsioonid ja andis raamistiku selle arendamiseks.

Pärast kirjelduse loomist alustati veebirakenduse minimaalse elujõulise toote *MVP* versiooni väljatöötamisega. See versioon sisaldab veebirakenduse põhifunktsioone.

Testimisfaasis saadud tagasisidet kasutati parendusvaldkondade väljaselgitamiseks[2].

Kogu arendusprotsessi vältel küsiti aktiivselt kliendilt tagasisidet. Seda tagasisidet on kasutatud veebirakenduse pidevaks täiustamiseks ning selle vastavuse tagamiseks kliendi vajadustele ja ootustele.

2 Olemasolevad rakendused

Selles peatükis uurib autor olemasolevaid veebirakendusi, mis on välja töötatud juuksurisalongi broneerimiseks. Autor analüüsib iga rakenduse omadusi, tugevusi ja nõrkusi. See ülevaade annab väärtuslikku teavet autori enda juuksuritöökoja broneerimise veebirakenduse arendamiseks.

Analüüsiks valis autor kolm juba olemasolevat rakendust:

- HairFcker – https://www.hairfcker.ee/
- Lumberjack – https://lumberjackbarbershop.com/
- Kvartals – https://www.kvartals.ee/

2.1 HairFcker

Hairfckeri veebisaidile minnes tervitab kasutajaid avaleht koos keritavate fotodega juuksuritöökoja ruumidest. Ülal on mitu navigeerimisnuppu, millest igaüks viib erinevale lehele.

Saidil on eraldi leht hindade loeteluga, mis on pluss. Otse nimekirjast aga aega broneerida ei saa, mida võib pidada oluliseks puuduseks. Lisaks on kogu kontaktandmed paigutatud eraldi lehele, mis võib kasutajatele ebamugavusi tekitada.

Üldjoontes võiks kodulehe visuaalset külge parandada, kuid disainis jääb puudu kasutatavusest ja ülevaatlikkusest.

2.2 Lumberjack

Lumberjacki saidil on vapustav disain, mis tõmbab saabumisel kohe silma. Saidi päises on palju navigeerimisnuppe, mis suunavad erinevatele lehtedele, kus on lisateavet, näiteks juuksuritöökoja ajalugu ja juuksurigalerii. Kodulehel on videod klientide soengu tegemisest, juuksuritöökoja lühikirjeldus ja sooviavaldus.

Broneerimisprotsessi alustamiseks peavad kasutajad klõpsama paremas alanurgas oleval ringil, mida võib olla raske näha, kuna see sulandub taustale. Broneerimisprotsess ise toimub paremal pool asuvas aknas, kus on võimalik valida töötaja, teenus ja aeg.

Üks negatiivne külg on see, et kasutajad peavad valima aja enne vabade töötajate kuvamist, mis muudab selle ebamugavaks neile, kes soovivad reitingute põhjal juuksurit valida.

Kuigi broneeringu saab teha ilma loata, on volitatud kasutajatel juurdepääs lisafunktsioonidele, nagu varasemate broneeringute vaatamine ja võimalus tulevasi broneeringuid tühistada. Üldiselt on veebisaidil visuaalselt atraktiivne disain, hästi organiseeritud teave ja palju animatsioone.

2.3 Kvartals

Selle veebisaidi avaleht on täis kasulikku teavet ja navigeerimisnuppe. Kasutajad leiavad hõlpsalt kontaktandmed, aadressi ja broneerimisnupu, mis on oluline eelis, kuna kogu vajalik teave on ühel lehel, mis teeb kohtumise kohese broneerimise lihtsaks. Kuid suutmatus saidi erinevatele keeltele vahetada on ilmne puudus.

Niipea kui kasutaja klõpsab nupul "E-BRONEERING", suunatakse ta teisele lehele, kus saab valida teenuse, aja ja eelistatud juuksuri. See on ka koht, kus kasutajad saavad saidi keelt vahetada.

Üks puudus, mida tuleb arvesse võtta, on saidi registreerimisprotsess, mis võimaldab kasutajatel registreeruda ainult teenuse Google+ või Facebooki kaudu. Traditsioonilist registreerimisvõimalust pole.

2.4 Olemasolevate rakenduste analüüs

Selles peatükis analüüsis autor kolme olemasolevat juuksurisalongi broneerimiseks välja töötatud veebirakendust: HairFcker, Lumberjack ja Kvartals.

HairFckeril puudub disainis kasutatavus ja lakoonilisus, samas kui Lumberjackil on visuaalselt atraktiivne disain, hästi organiseeritud teave ja palju animatsioone. Broneerimisprotsessil on aga üks puudus: kasutajad peavad valima aja, enne kui kuvatakse vabad töötajad. Kvartals pakub märkimisväärset eelist, kuna kogu vajalik teave on ühel lehel, mis muudab kohtumise kohe broneerimise lihtsaks. Kuid suutmatus saidi erinevatele keeltele vahetada on ilmne puudus ja registreerimisprotsess võimaldab kasutajatel registreeruda ainult Google+ või Facebooki kaudu, ilma traditsioonilise registreerimisvõimaluseta.

Kokkuvõttes annab see ülevaade autorile väärtuslikku teavet oma juuksuripoe broneerimise veebirakenduse väljatöötamiseks, võimaldades autoril ära kasutada olemasolevate rakenduste tugevaid külgi ja vältida nende puudusi.

3 Tehnoloogiad

Selles peatükis räägib autor tehnoloogiatest, mida selle projekti väljatöötamisel kasutatakse. Tehnoloogia on iga tarkvaraarendusprojekti kriitiline komponent, kuna see võib oluliselt mõjutada lõpptoote funktsionaalsust, jõudlust ja mastaapsust. Tõhusa ja tõhusa lahenduse loomiseks on oluline hoolikalt läbi mõelda, milliseid tehnoloogiaid kasutada ja kuidas neid rakendada.

3.1 Nõuded

Selles alapeatükis kirjeldatakse kasutajalugudena funktsionaalseid ja mittefunktsionaalseid nõudeid, millele juuksuritöökoja veebirakendus peab vastama. Peamised kasutajarollid on juuksur, administraator ja klient.

Funktsionaalsed nõuded klientidele:

- Kliendina soovin, et oleks võimalik aeg internetis kokku leppida.
- Kliendina soovin kliendina näha oma eelistatud juuksuri saadavust.
- Kliendina soovin, et mul oleks võimalik kohtumisi tühistada või ümber ajada.
- Kliendina soovin näha juuksuri pakutavate erinevate teenuste hinnakirja.

Mittefunktsionaalsed nõuded klientidele:

- Kliendina soovin, et veebirakendus oleks kasutajasõbralik ja hõlpsasti navigeeritav.
- Kliendina soovin, et veebirakendus oleks kiire ja tundlik.
- Kliendina soovin, et veebirakendus oleks juurdepääsetav mitmest seadmest.

Funktsionaalsed nõuded juuksuritele:

- Juuksurina tahan näha oma päevaseid kohtumisi.
- Juuksurina tahan näha oma päevast tööaega.
- Juuksurina tahan näha kohtumise üksikasju.
- Juuksurina tahan näha oma kolleege kohtumas.

Mittefunktsionaalsed nõuded juuksuritele:

- Juuksurina soovin, et veebirakendus oleks turvaline ja kaitseks minu isikuandmeid.
- Juuksurina soovin, et veebirakendus oleks töökindel ja kättesaadav 24/7.
- Juuksurina soovin, et veebirakendus oleks skaleeritav ja saaks hakkama suure hulga kohtumistega.

Funktsionaalsed nõuded administraatorile:

- Administraatorina soovin, et veebirakendus saaks süsteemist juuksureid lisada, redigeerida ja eemaldada.
- Administraatorina soovin, et veebirakendus saaks hallata juuksuri ajakava ja saadavust.
- Administraatorina soovin, et veebirakendus saaks vaadata ja hallata klientide kohtumisi.

Mittefunktsionaalsed administraatorinõuded:

- Administraatorina soovin, et veebirakendusel oleksid turvameetmed klientide ja ettevõtete andmete kaitsmiseks.
- Administraatorina soovin, et veebirakendusel oleks kasutajasõbralik liides, mis hõlbustaks kasutamist ja kiiret navigeerimist.
- Administraatorina soovin, et veebirakendusel oleks tundlik disain, et seda saaks kasutada erinevatel seadmetel ja ekraanisuurustel.

- Administraatorina soovin, et veebirakendus oleks väga kättesaadav ja töökindel, et minimeerida seisakuid.
- Administraatorina soovin, et veebirakendus oleks skaleeritav, et mahutada kliendi- ja äriandmete kasvu.
- Administraatorina soovin, et veebirakendusel oleks mitmekeelsete klientide mitmekeelne tugi.

Ajapiirangu tõttu jäeti sellest lõputööst välja mõned planeeritud funktsioonid, nagu klientide tagasiside andmise võimalus, samuti võimalus kohtumisi automaatselt mainida. Need funktsioonid vaadatakse üle ja arendatakse tulevikus.

3.2 Veebirakendus

Juuksuritöökoja broneerimissüsteemi jaoks valiti juurutamise platvormiks veebirakendus selle kasutajasõbralikkuse tõttu. Veebirakenduste üks peamisi eeliseid on nende kasutuslihtsus, kuna kasutajad pääsevad süsteemile ligi lihtsalt brauseri ja Interneti-ühenduse kaudu, ilma et oleks vaja alla laadida või värskendusi[4]. Lisaks on lai valik programmeerimiskeeli ja veebirakenduste arenduskeskkondi, mis pakuvad suurt arenduspaindlikkust[3].

Veebirakenduste teine oluline eelis on see, et kasutajaandmed salvestatakse serverisse, mis muudab need pärast saidil autoriseerimist hõlpsasti kättesaadavaks mis tahes seadmest[5]. See tähendab, et kliendid saavad oma tellimuste ajalugu vaadata või uusi kohtumisi kokku leppida mis tahes Interneti-juurdepääsuga seadmest.

Veebirakendustel on aga ka omad miinused. Üks negatiivne külg on nende haavatavus turvarikkumiste suhtes, kuna häkkimised ja identiteedivargused kasvavad jätkuvalt. Lisaks on veebirakendustel sageli kehv jõudlus võrreldes muud tüüpi rakendustega, mis saavad arenduse käigus platvormipõhiseid funktsioone täielikult ära kasutada[6].

Nendele puudustele vaatamata on veebirakendused paljudele ettevõtetele kulutõhus ja praktiline valik, kuna neid on lihtne juurutada ja hooldada mitmes seadmes[7]. Üldiselt otsustati kasutada veebipõhist juuksurisalongi broneerimissüsteemi pärast projekti eeliste ja puuduste ning ka nõuete hoolikat kaalumist.

3.3 Arenduskeelte valik

C# on taustaprogrammi arendamiseks populaarne valik, kuna seda on lihtne õppida ja kasutada, sellel on tugev tugi tarkvara kujundamise mustritele ja see võib töötada mitmel platvormil[8]. See on väga populaarne ka veebirakenduste arendamiseks, kuigi on ka teisi keeli, mida tavaliselt kasutatakse selleks otstarbeks, näiteks PHP, Python ja JavaScript.

Võrreldes PHP-ga on C#-l tugevam tüüpi süsteem ja see on rohkem keskendunud objektorienteeritud programmeerimisele, muutes tugeva ja hooldatava koodi kirjutamise lihtsamaks. C# toetab paremini ka tarkvara kujundamise mustreid[19].

Python on veel üks populaarne veebiarenduskeel ja kuigi sellel on lihtsam süntaks kui C#, puudub sellel C# tugev tüübisüsteem ja jõudluse optimeerimine. Pythoni tugevused seisnevad selle mitmekülgsuses, kasutuslihtsuses ja suures kolmandate osapoolte pakettide raamatukogus.

JavaScripti, mida kasutatakse peamiselt esiotsa veebiarenduseks, saab Node.js-iga kasutada ka sisekeelena. Kuigi JavaScripti eeliseks on võimalus kasutada sama keelt nii esi- kui ka tagaotsa arendamiseks, võib see olla vähem toimiv kui C# ja nõuda täiendavaid optimeerimispüüdlusi[20].

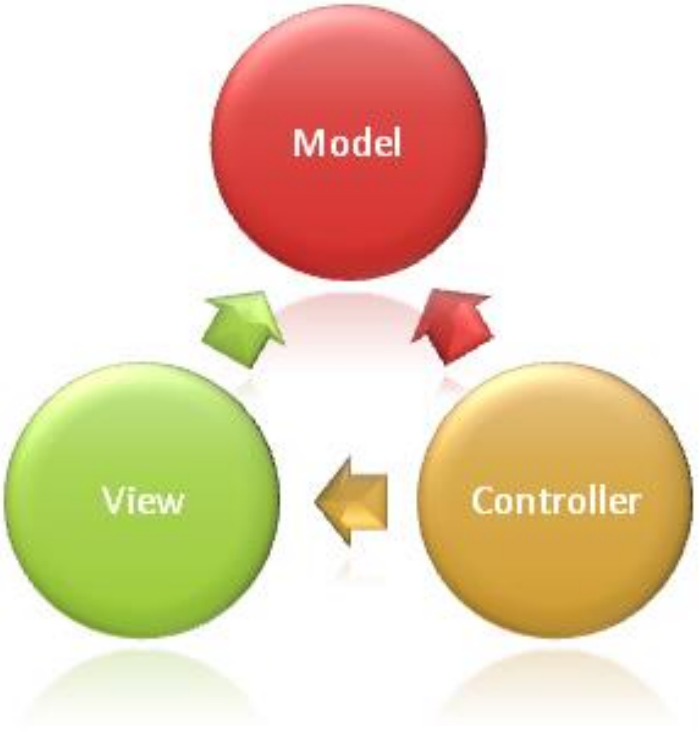
C# kasutamise üks peamisi eeliseid veebirakenduste arendamiseks on selle integreerimine Microsoft *.NET* platvormiga. *.NET* platvorm pakub rikkalikke teeke ja tööriistu veebirakenduste arendamiseks, sealhulgas veebivormid, *MVC* (Model-View-Controller) arhitektuur ja veebi-*API* (rakenduse programmeerimisliides) *RESTful* teenuste loomiseks[9]. Lisaks toetab C# laia valikut kolmandate osapoolte teeke ja raamistikke, näiteks objektide relatsioonilise kaardistamise jaoks Entity Framework ja reaalajas suhtlemine SignalR.

Lõpuks sõltub veebirakenduse sisekeele valik paljudest teguritest, nagu projekti spetsiifilised nõuded, arendaja kogemused ja teadmised, raamatukogude ja tööriistade saadavus. Käesoleva töö autor valis C# eelkõige seetõttu, et ta on seda programmeerimiskeelt kõige paremini kursis ning ka selle integratsiooni tõttu .NET platvormiga ja tugeva tüüpi süsteemiga.

ASP.NET Core MVC on võimas ja paindlik raamistik veebirakenduste arendamiseks, kasutades *MVC* arhitektuurimustrit. *MVC* abil saate luua oma veebirakendusele dünaamilisi, interaktiivseid ja reageerivaid kasutajaliideseid. Platvorm on üles ehitatud *.NET Core* käitusajale, mis pakub rikkalikku tööriistade ja teekide komplekti suure jõudlusega ja skaleeritavate veebirakenduste loomiseks[10].

Üks *ASP.NET Core MVC* tugevusi on selle võime eraldada esitluskiht (vaade) äriloogikast ja andmete juurdepääsukihist (mudel ja kontroller) (Joonis 1). See eraldamine võimaldab teil luua mugavamat, testitavamat ja laiendatavamat koodi. Lisaks pakub platvorm laia valikut funktsioone ja tööriistu kaasaegsete veebirakenduste loomiseks, sealhulgas marsruutimine, andmete valideerimine, autentimine ja autoriseerimine[10].

Kokkuvõttes on *ASP.NET Core MVC* tugev ja mitmekülgne platvorm, mis aitab teil hõlpsalt veebirakendust luua. Selle paindlik arhitektuur koos rikkalike funktsioonide ja tööriistadega muudab selle ideaalseks valikuks juuksuritöökoja broneerimissüsteemi arendamiseks.



Joonis 1. MVC arhitektuuri struktuur

3.4 Andmebaasi valik

Autor otsustas veebirakenduse andmebaasina kasutada PostgreSQL-i, kuna see on populaarne avatud lähtekoodiga O*RDBMS*, mida kasutatakse laialdaselt veebirakenduste arendamiseks. Võrreldes teiste populaarsete andmebaasidega, nagu MySQL ja MongoDB, on PostgreSQL-il mitmeid eeliseid, mis muudavad selle veebirakenduste jaoks suurepäraseks valikuks. Sellel on rikkalik funktsioonide komplekt, sealhulgas täiustatud andmetüüpide tugi, võimsad indekseerimisvõimalused ja põhjalik tehinguhaldus[11].

Üks PostgreSQL-i peamisi eeliseid on selle täiustatud *SQL*-funktsioonide ja keeruliste päringute tugi. PostgreSQL toetab laia valikut andmetüüpe, sealhulgas massiive, hstore ja *JSON*, muutes keerukate andmestruktuuridega töötamise lihtsaks. Samuti toetab see täiustatud indekseerimismeetodeid, nagu täistekstiotsing, ning pakub tööriistu päringu analüüsiks ja optimeerimiseks.

Teine PostgreSQL-i eelis on selle töökindlus ja andmete terviklikkus. PostgreSQL kasutab *MVCC*, mis tagab, et tehinguid täidetakse järjepidevalt ja andmed ei lähe kaduma ega rikuta samaaegse juurdepääsu tõttu. Samuti toetab see selliseid funktsioone nagu päästikud ja piirangud, mis võimaldavad arendajatel ärireegleid jõustada ja andmete järjepidevust tagada.

PostgreSQL toetab tugevalt ka mastaapsust ja jõudlust. See toetab päringu paralleelset täitmist ja suudab hõlpsasti töödelda suuri andmemahtusid. Samuti pakub see tööriistu andmebaasi jõudluse jälgimiseks ja optimeerimiseks.

Võrreldes MySQL-iga on PostgreSQL-il tugevam funktsioonide komplekt ja see sobib paremini keerukate andmemudelite ja päringute jaoks[21]. Kuigi MySQL-i on lihtsam seadistada ja kasutada, puuduvad sellel paljud PostgreSQL-i täiustatud funktsioonid.

Võrreldes MongoDB-ga, populaarse dokumendipõhise NoSQL-i andmebaasiga, pakub PostgreSQL struktureeritumat ja organiseeritumat andmehoidlat ning paremat tuge keerukate päringute ja tehingute jaoks[12]. MongoDB sobib paremini paindliku skeemiga struktureerimata andmete jaoks, PostgreSQL aga fikseeritud skeemiga struktureeritud andmete jaoks.

Lisaks on PostgreSQL-il suur ja aktiivne kogukond ning tihe integratsioon teiste veebiarenduses tavaliselt kasutatavate tehnoloogiatega, nagu *ASP.NET Core MVC*.

4 Lahenduse realiseerimine

Selles osas kirjeldab autor veebirakenduse juurutamise protsessi, mis hõlmab rakenduse arhitektuuri, andmebaasi, taustaosa ja esiosa väljatöötamist. Rakendusetapp on projekti oluline osa, kuna see hõlmab rakenduse disaini ja planeerimise muutmist funktsionaalseks süsteemiks, mida saab testida ja täiustada. Autor kirjeldab üksikasjalikult rakenduse arendamiseks tehtud samme, sealhulgas kasutatud tehnoloogiaid.

4.2 Arendusmetoodika

Projekti elluviimise faasis otsustas autor kasutada Agiilset arendusmetoodikat koos iganädalaste sprintidega, et tagada arendusprotsessi süsteemne ja organiseeritud lähenemine[13].

Autor alustas arendusprotsessi, jagades projekti nõuded väikesteks juhitavateks ülesanneteks. Seejärel jagati need ülesanded iganädalasteks sprindideks, millest igaühel olid kindlad eesmärgid ja eesmärgid, mida tuleb saavutada.

Nädala jooksul töötas autor määratud ülesannetega ja jälgis regulaarselt edenemist. See võimaldas autoril teha vajalikke kohandusi ja täiustusi ning tagas, et projekt kulgeb plaanipäraselt.

Tõhusa suhtluse ja koostöö tagamiseks arendusprotsessi käigus teavitas autor regulaarselt kliente töö edenemisest ning küsis nende tagasisidet ja arvamust. See võimaldas autoril oma ideid ja ettepanekuid projekti kaasata, tagades, et lõpptoode vastab nende ootustele ja nõuetele.

Üldiselt on agiilse arendusmetoodika kasutamine iganädalaste sprintidega aidanud autoril rakenduse arendamisel olla organiseeritud ja tõhus. See võimaldas autoril oma aega ja ressursse tõhusalt hallata, tagas autori edenemise projekti lõpuleviimise suunas ning tõhusa suhtluse ja koostöö kliendiga.

4.3 Andmebaasi realiseerimine

Andmebaasi juurutamine selle projekti jaoks oli kriitiline ja autor kulutas palju aega erinevate võimaluste uurimisele ja testimisele, enne kui otsustas kasutada andmebaasihaldussüsteemina PostgreSQL-i. PostgreSQL on tuntud oma töökindluse, tõrketaluvuse ja mastaapsuse poolest, mistõttu on see selle projekti jaoks suurepärane valik[11].

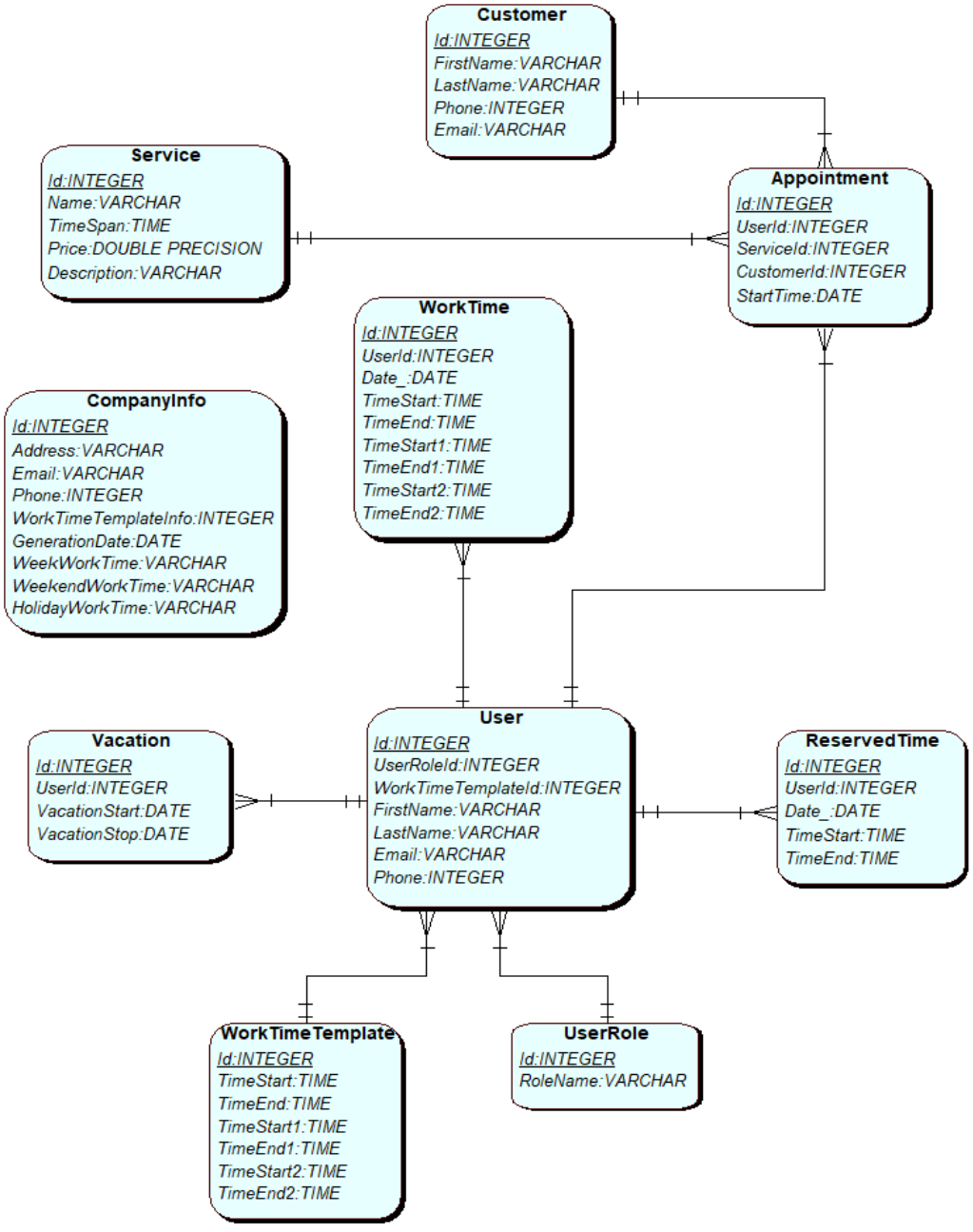
PostgreSQL-iga töötamiseks ja andmebaaside migratsiooni haldamiseks otsustas autor kasutada Entity Frameworki, mis on .NET-i jaoks populaarne Object-Relational Mapping *ORM* raamistik[14]. Entity Frameworki abil sai autor C# koodi abil määrata andmebaasi skeemi ja kaardistada selle PostgreSQL-is vastavate andmebaasi tabelitega.

Entity Frameworki kasutamise üks peamisi eeliseid on see, et see pakub andmebaasi arendamisel koodipõhist lähenemist, võimaldades autoril keskenduda rakenduse andmemudeli määratlemisele, mitte muretseda aluseks oleva andmebaasi skeemi pärast. See lähenemisviis hõlbustas ka koodibaasi haldamist ja vajaduse korral andmebaasi skeemi värskendamist.

Veebi rakenduse andmebaas koosneb 10st tabelist (Joonis 2). Kõike rakenduse spetsiifilisi tabeleid hoitakse ühes skeemis. Iga keskkonna jaoks on loodud eraldi iseseisev andmebaasi instants. Rakenduse tabeliteks on:

- User – Sisaldab kõiki registreeritud kasutajaid. Sisaldab registreeritud tarbijaid ja administraatori lisatud kasutajaid.
- UserRole – sisaldab „User“ rolli. Kui klient registreerib, omistatakse talle automaatselt „Customer“ roll. Administraatori poolt loomisel saab rolli valida.
- CompanyInfo – sisaldab kogu ettevõttega seotud kasulikku teavet.
- Vacation – sisaldab "User" puhkuse algus- ja lõppkuupäeva.
- WorkTime – sisaldab töötajale kuni 3 intervalli töötundi päevas.
- WorkTimeTemplate - sisaldab kuni 3 tööaja intervalli edasiseks automaatseks tööaja genereerimiseks.
- Service – sisaldab juuksuri pakutavate teenuste loetelu.

- Customer – aja broneerimisel sisaldab registreerimata kasutajate kontaktandmeid.
- Appointment – sisaldab kogu teavet broneeringu kohta.
- ReservedTime – sisaldab töötajale broneeritud aega.



Joonis 2. Rakenduse andmemudelid

4.4 Tagasüsteemi arhitektuur

Rakenduse taustaprogrammi on rakendatud .NET-raamistiku uusima versiooni, täpsemalt .NET 7.0 abil. Selle platvormi valik põhines mitmel teguril, sealhulgas selle töökindlusel, mastaapsusel ja laialdasel kasutusel selles valdkonnas. Mall koosneb kolmest põhikihist, millele on projekti jaoks lisatud valikuline neljas kiht.

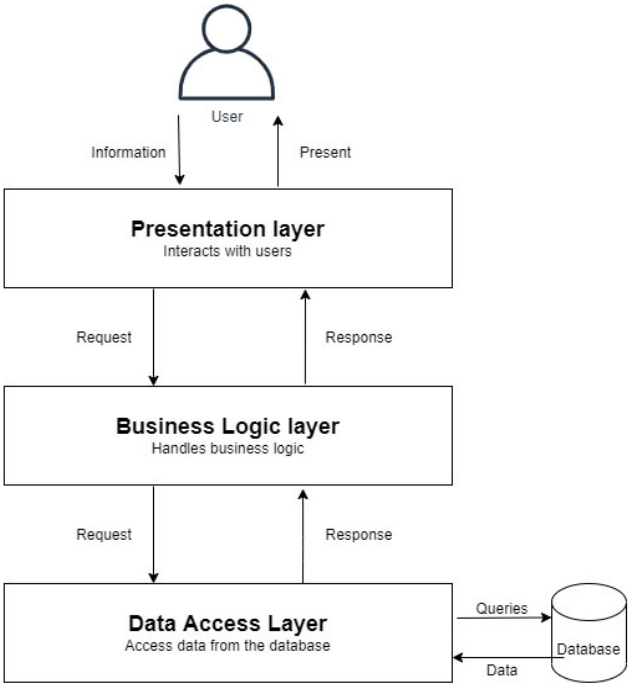
See projekt kasutab three-layer arhitektuuri. Three-layer architecture arhitektuur on tarkvara kujundamise muster, mida kasutatakse veebirakenduste arendamiseks[15]. See koosneb kolmest tasemest, millest igaühel on oma spetsiifilised kohustused (Joonis 3):

Esitluskiht (või kasutajaliides): see kiht vastutab andmete kasutajale kasutajasõbraliku esitamise eest. See sisaldab kõike, millega kasutaja suhtleb, näiteks vorme, nuppe ja menüüsid.

Ärikiht (või rakenduseloogika): see kiht sisaldab rakenduse põhifunktsioone, nagu ärireeglid, töövood ja andmete valideerimine. Ta vastutab andmete töötlemise ning nende õigsuse ja järjepidevuse eest.

Andmekiht (või püsivuskiht): see kiht vastutab andmete salvestamise ja andmebaasist toomise eest. See hõlmab kõike andmetele juurdepääsuga seonduvat, näiteks ühendusi, tehinguid ja päringuid.

Kolmetasandiline arhitektuur eraldab rakenduse eraldi kihtideks, muutes selle haldamise ja hooldamise lihtsamaks. See tagab ka parema skaleeritavuse ja paindlikkuse, kuna ühes kihis tehtud muudatused ei mõjuta teisi kihte.



Joonis 3. Kolmekihiline arhitektuur

4.3.1 Tagasüsteemi realiseerimine

Taustaarhitektuuri esimene kiht on Data Access Layer, mis tegeleb kõigi andmebaasiga seotud ülesannetega. See tase hõlmab andmebaasi olemi sätteid, andmebaasireegleid ja andmebaasi olemi seoseid. *DAL* suhtleb andmebaasiga ja hangib või värskendab sealt andmeid[15].

Äriobjektide kiht sisaldab objekte, mida rakenduses kasutatakse, ja tavalisi abifunktsioone (ilma loogikata), mida kõik kihid kasutavad. Kolmetasandilises arhitektuuris on see kiht valikuline. Kuna aga autor järgib OOP-i C#-ga, siis tuleb korduv kood hoida võimalikult lühike. Seetõttu on oluline kasutada kihti tavaliste koodide salvestamiseks, mitte neid igasse kihti salvestada.

Selle kihi loomiseks kasutatakse Entity Framework teeki, mis vähendab oluliselt sõltuvust rakenduse ja andmebaasi vahel ning lihtsustab oluliselt andmevahetust. Lisaks võimaldab Entity Framework kasutada LINQ-tehnoloogiat andmebaasipäringute jaoks, mida saab puhta SQL-i asemel kirjutada C#-s, mis muudab koodi loetavamaks ja aitab vigu kiiremini tabada[15].

Sisearhitektuuri teine kiht on äriloogikakiht *BLL*, mis sisaldab kogu rakenduse äriloogikat. See kiht töötleb *DAL*-ist saadud andmeid ja teeb kõik vajalikud arvutused, kontrollid või manipulatsioonid. *BLL* suhtleb *DAL*-iga, et vajadusel andmeid hankida või värskendada[15].

Sisearhitektuuri kolmas ja viimane kiht on veebirakenduste kiht *WAL*, mis hakkab tegelema kõigi kasutajaliidesega seotud ülesannetega. See kiht sisaldab veebilehti, veebiteenuseid ja muid kasutajaga suhtlemiseks vajalikke komponente. *WAL* suhtleb *BLL*-iga andmete toomiseks või värskendamiseks ning kasutaja sisendi põhjal vajalike toimingute tegemiseks[15].

Lõpuks lisati arhitektuurile täiendav kiht, mis on projekti jaoks vajalik. See viies tase koosneb ressursifailidest, mida kasutatakse mitmesuguste keelte tõlgete salvestamiseks, mida veebirakendus peab toetama. Tõlked ja neile vastavad kultuuritüübid registreeritakse rakenduse käivitamisel ja märgitakse vaatemudelite atribuudiks ning vaadete jaoks lisatakse tõlge otse Razori süntaksi abil.

Üldiselt eraldab see kolmetasandiline arhitektuur mured selgelt, muutes koodi modulaarsemaks, hooldatavamaks ja testitavamaks. See hõlbustab ka skaleerimist, kuna iga taset saab rakenduse nõuete alusel iseseisvalt skaleerida. Lõpuks tagab .NET 7.0 kasutamine, et serveri arhitektuur on kiire, turvaline ja töökindel.

4.4 Esirakenduse arendus

Käesolevas lõputöös kasutas autor rakenduse esitluskihi loomiseks *ASP.NET Core MVC*-d. Esitluskiht vastutab kasutajaliidese pakkumise eest rakendusega suhtlemiseks. See kiht koosneb vaatemudelitest, vaadetest ja kontrolleritest.

Vaatemudelid määravad vaadetes kasutatavad andmestruktuurid ning vaated *HTML*-mallid ja kasutajaliidese. Kontrollerid töötlevad kasutaja sisendit ja korraldavad suhtlust esitluskihi ja äriloogikakihi vahel.

MVC-kontrollerite loomise veelgi lihtsustamiseks on mõned kontrollerid loodud ASP.NET CodeGenerator kontrolleri abil. Autor kasutas responsiivsete ja dünaamiliste veebilehtede loomiseks *HTML*-i, *CSS*-i ja JavaScripti.

Arendusprotsessi hõlbustamiseks ning rakenduse ühtse välimuse ja tunnetuse tagamiseks kasutas autor raamistikku Bootstrap 5. See raamistik annab valmis komponentide, *CSS*-klasside ja JavaScripti funktsioonide komplekti, mida saab hõlpsasti veebilehtedesse integreerida.

Rakendusele interaktiivsuse lisamiseks kasutas autor jQuery teeki[16]. See teek pakub lihtsat ja mugavat liidest *HTML*-dokumendi elementidega manipuleerimiseks ja asünkroonsete *HTTP*-päringute tegemiseks.

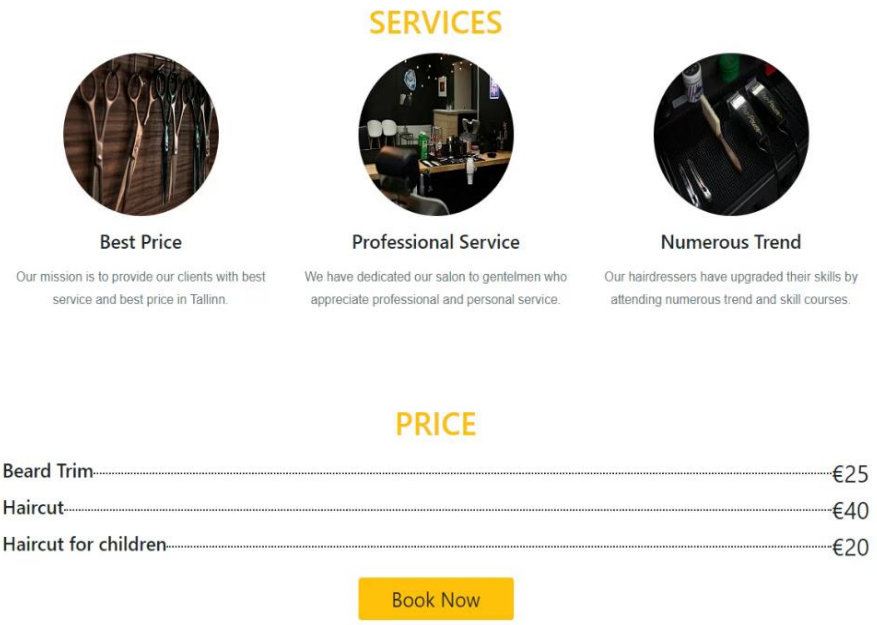
4.5 Rakenduse funktsionaalsus

Selles osas räägime veebirakenduse funktsionaalsusest aja broneerimiseks. See peatükk tutvustab funktsioone klientidele, juuksuritele ja administraatoritele.

4.5.1 Koduleht registreerimata kasutajatele

Sellel veebirakendusel on võimalus saada mõningaid funktsioone isegi registreerimata kasutajatele. Seda tehakse seetõttu, et paljud juuksuritöökoja kasutajad võivad olla lihtsalt juhuslikud kliendid, kes ei kavatse teenuseid rohkem kui üks kord kasutada. Nende jaoks on rakendatud võimalus broneerida aeg isegi registreerumata.

Avalehel näete mõnda fotot, ettevõtte teavet ja kontaktteavet, nagu aadress, telefoninumber ja lahtiolekuajad. Avalehel on ka teenuste nimekiri koos hindadega, mida pakub juuksur. (Joonis 4) Nii ei pea kasutaja hinnakirja vaatamiseks teistele lehekülgedele minema ega isegi registreeruma.

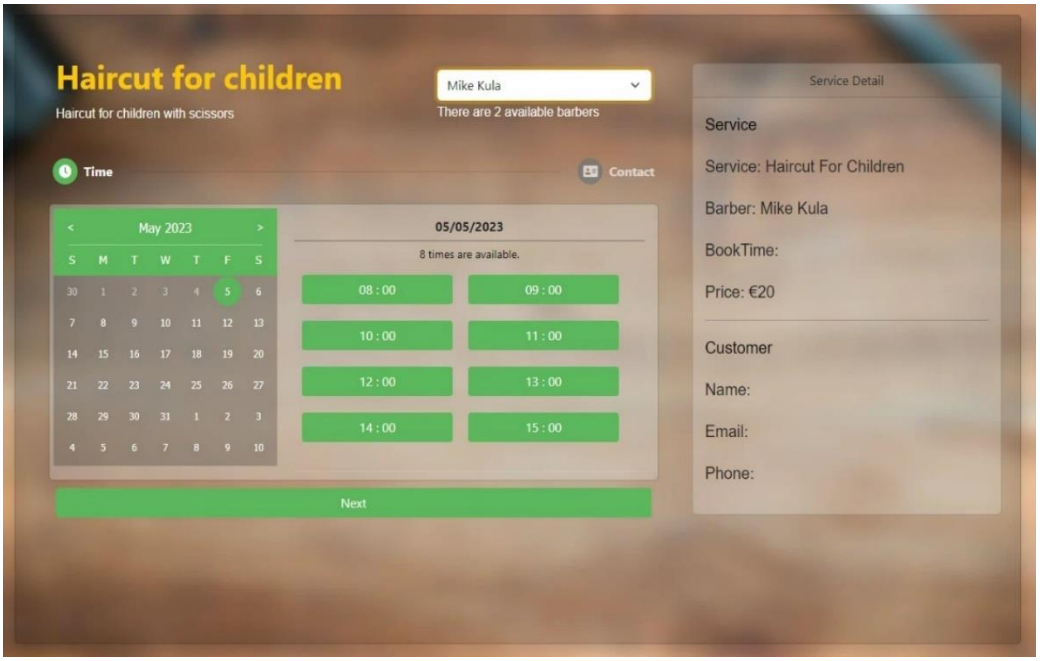


Joonis 4. Hinnakiri avalehel

4.5.2 Aja broneerimise protsess

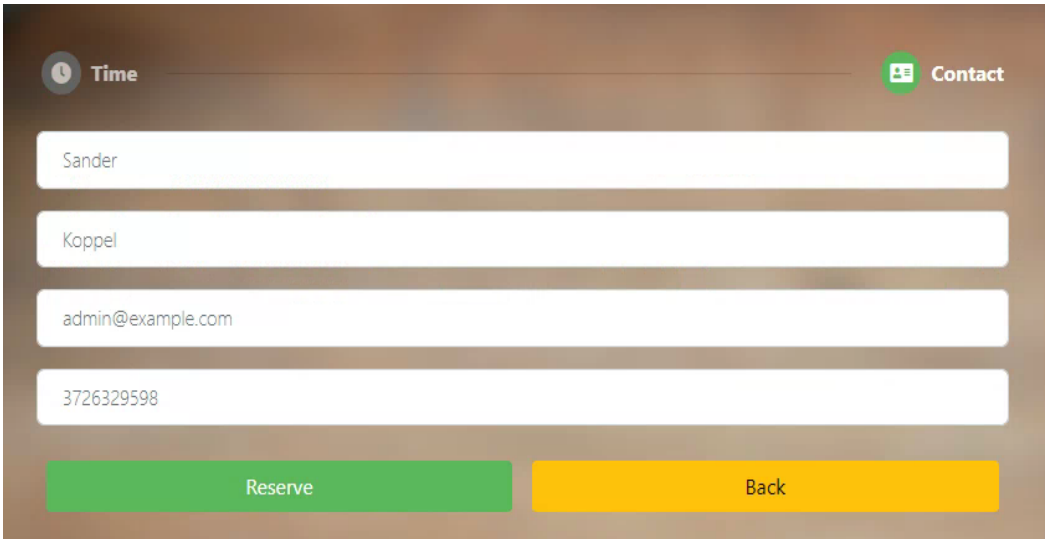
Klõpsates nupul “Book Now” (Joonis 4), suunatakse kasutaja lehele, kus on juuksuri pakutavate teenuste loetelu. Kasutaja saab valida nimekirjast, mida ta on avalehel juba näinud.

Konkreetse teenuse valikul suunatakse kasutaja aja broneerimise lehele (Joonis 5). Aja broneerimise lehel palutakse kasutajal valida meister ja kellaaeg



Joonis 5. Aja broneerimine

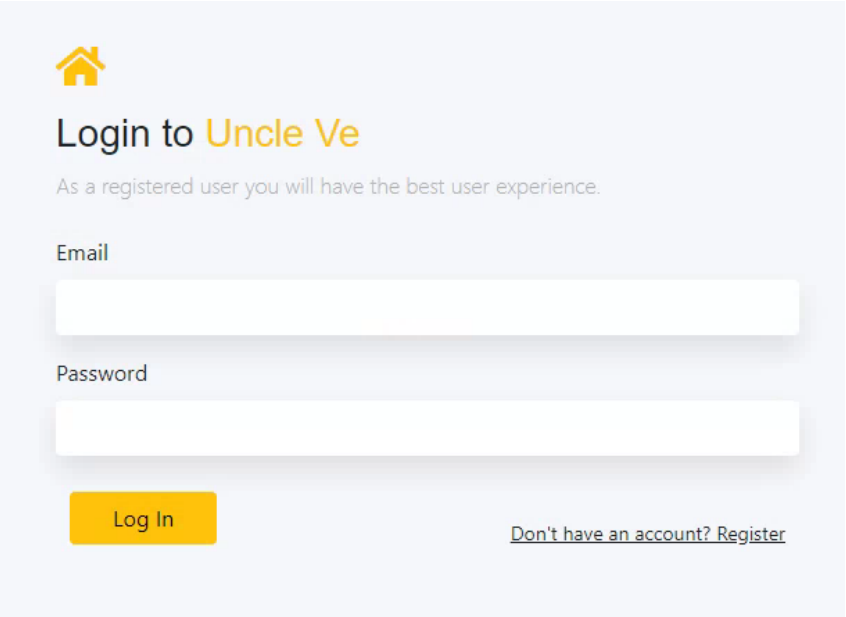
Aja valimisel palutakse registreerimata kasutajal sisestada oma kontaktandmed. Registreeritud kasutajate puhul jäetakse see samm vahele (Joonis 6).



Joonis 6. Kontaktandmete lisamine

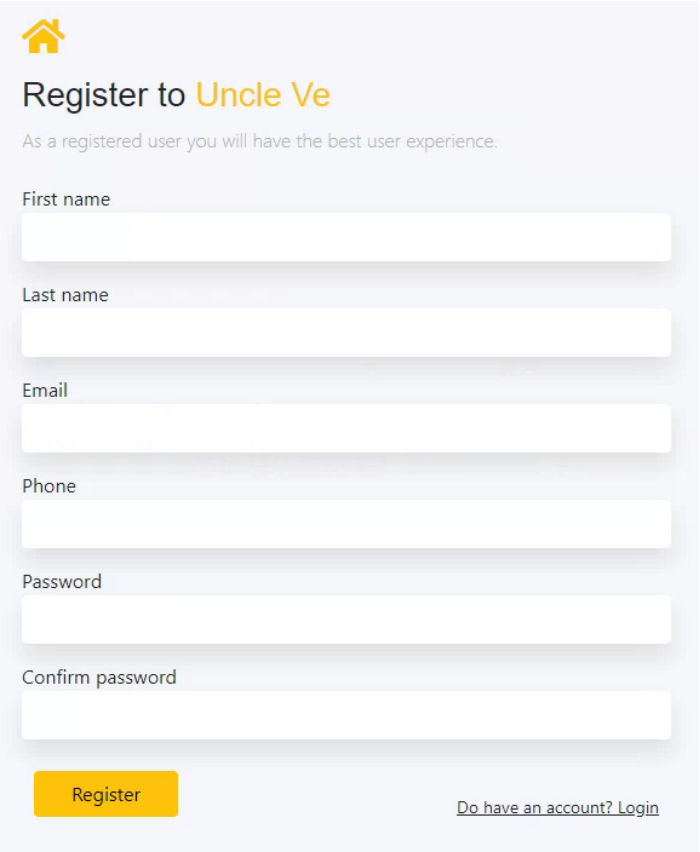
4.5.3 Sisselogimine ja registreerimine

Kasutajal on võimalus sisestada oma meiliaadress ja parool, mis saadetakse autentimiseks teeninduskihti (Joonis 7). Kui kasutaja andmed on õiged, volitatakse ta ja suunatakse veebilehe avalehele. Navigeerimismenüüsse ilmuvad lisafunktsioonid, nagu "Profiil" ja võimalus vaadata oma eelseisvaid juuksuriseansse, aga ka varasemaid broneeringuid.



Joonis 7. Sisselogimise vaade

Kasutaja registreerimisel tuleb sisestada oma kontaktandmed ja luua uuele kontole (Joonis 8) parool. Kui kõik täidetud väljad vastavad nõuetele ja antud e-mail pole veel registreeritud, siis toimub registreerimisprotsess. Kui selle e-kirjaga on kunagi olnud juukselõikuskirjeid, kantakse kõik need kanded äsja loodud kontole.

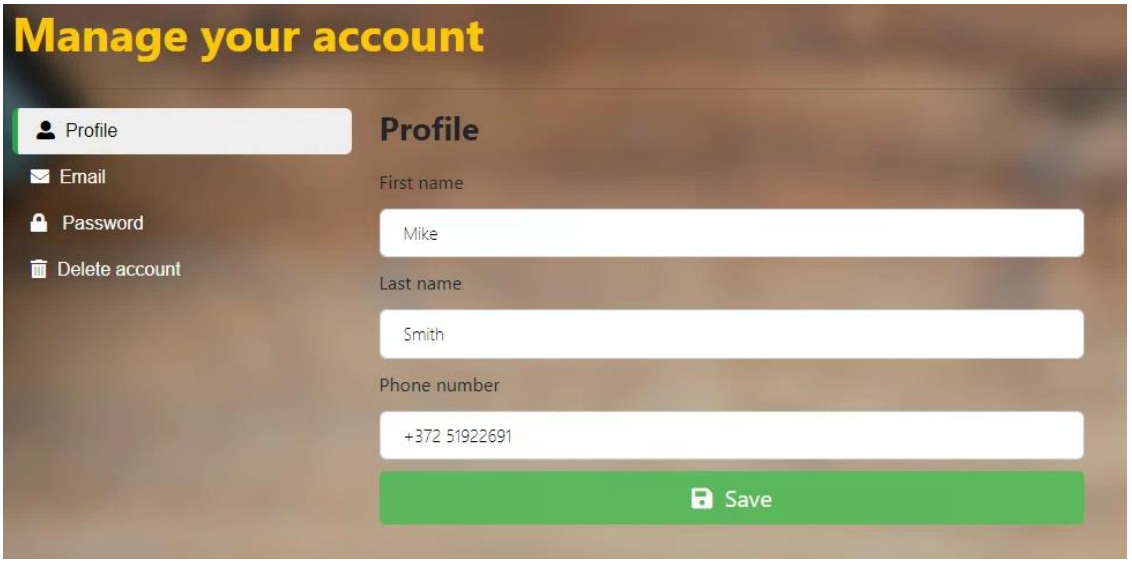


Joonis 8. Registreerimise vaade

4.5.4 Profiili redigeerimine

Registreeritud kasutajatele ilmuvad mitmed kasulikud funktsioonid (Joonis 9). Üks neist on võimalus oma profiili muuta. Profiiliredaktoris on võimalik muuta kõiki kontaktandmeid, sh meili. Samuti on võimalik parooli muuta.

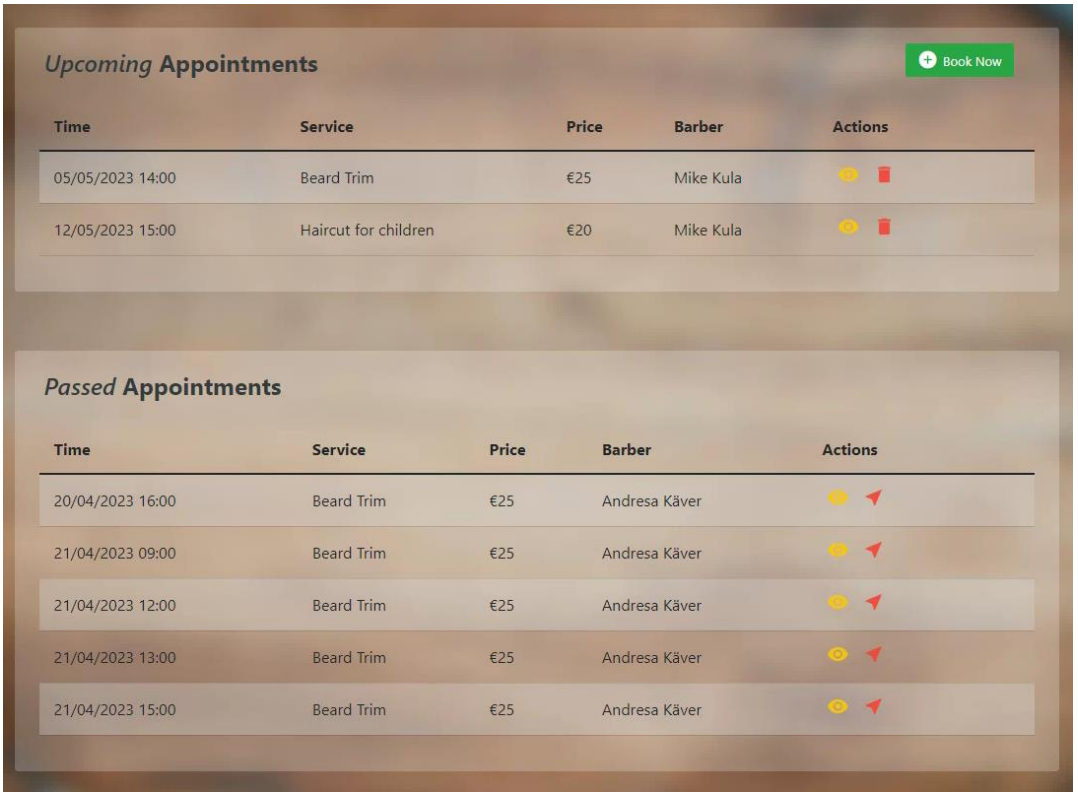
Soovi korral on kasutajal võimalus oma profiil täielikult kustutada. Need profiilifunktsioonid on samad nii klientidele kui ka töötajatele.



Joonis 9. Kasutajaprofiil

4.5.5 Tulevasi ja varasemaid broneeringuid

Registreeritud kasutajatel on võimalus vaadata oma praeguseid ja varasemaid broneeringuid. (Joonis 10) Kasutajal on võimalus vaadata lisainfot oma broneeritud aegade kohta, samuti on võimalik neist ükskõik milline tühistada. Varasemate kohtumiste jaoks on võimalik aeg uuesti broneerida. See tähendab, et kasutajal on võimalus klõpsata mis tahes varasemal kohtumisel ja ta suunatakse lehele, kus on juba valitud juuksur ja teenus.

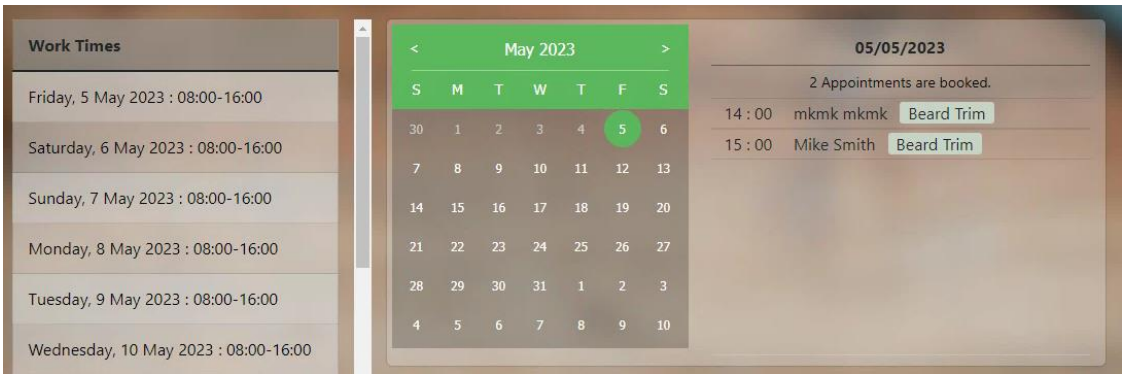


Joonis 10. Tulevasi ja varasemaid broneeringuid

4.5.6 Juuksuri graafikud

Töötajana on sul endal elu lihtsamaks tegemiseks mitu funktsiooni. Juuksuril on võimalus vaadata nii oma tööaega kui ka iga tööpäeva graafikut (Joonise 11). Samuti näete mis tahes broneeritud ajal klõpsates selle kohta lisateavet.

Eraldi vahekaardil, kuhu pääseb navigeerimisnupu kaudu, on kolleegide nimekiri. Samuti saab kasutaja vaadata kolleegide tööaegu ja ajakava, et nendevaheline paremini koordineerida.



Joonis 11. Juuksuri graafikud

4.5.7 Administraatori külgfunktsioonid

Administraatorina on teil palju võimalusi. Administraatoril on võimalus muuta lisamist ja kustutamist ning muuta teavet ettevõtte kohta, nagu aadress, telefon ja juuksuritööaeg.

Samuti on administraatoril võimalus lisada töötajatele vabu päevi, mida arvestatakse kõigi töötajate tööaja genereerimisel (Joonise 12).

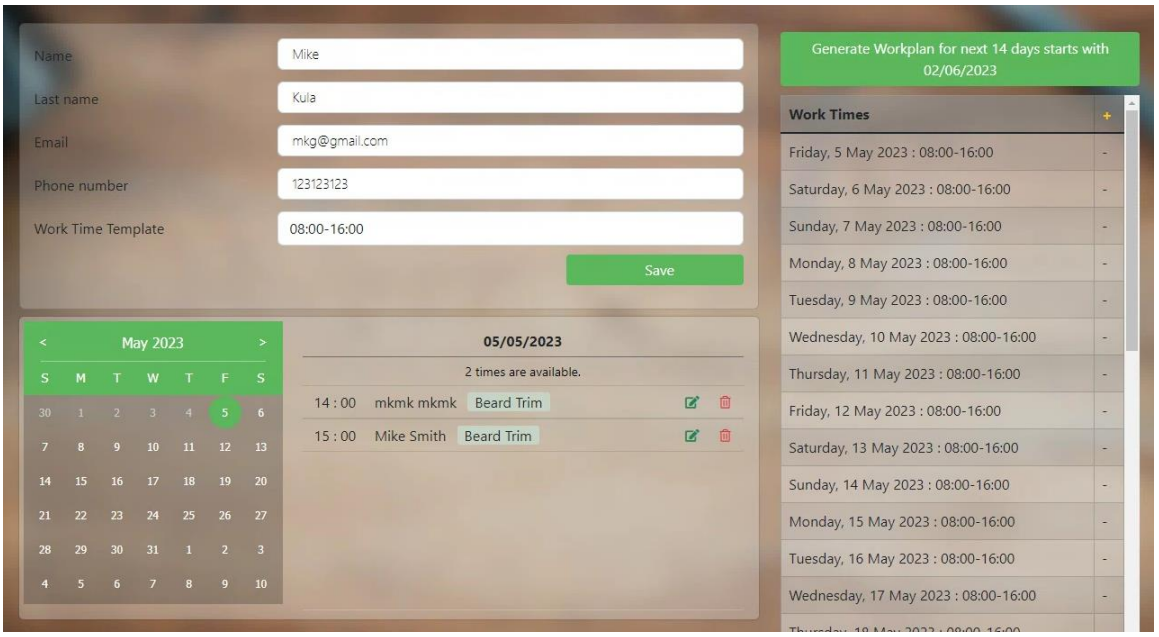
Joonis 12. Töötajatele vabad päevad

4.5.8 Administraatori põhifunktsioonid

Selles projektis rakendati palju erinevaid administraatori funktsioone. Administraatori põhifunktsioonide hulka kuulub võimalus registreerida uusi töötajaid ja neid kustutada.

Kuna kliendi juuksurisalong töötab nii, et osadel töötajatel on võimalus ise valida päevad ja kellaajad, millal ta soovib töötada. Selleks oli administraatoril võimalus lisada töötajale 3 töötundi päevas (Joonis 13)

Aga seal töötavad samad inimesed, kellel on stabiilne tööpäev. Selliste töötajate jaoks rakendati võimalust võtta vabu päevi, samuti määrata šabloon, et genereerida automaatselt tööajad teatud ajaks ette (Joonis 13).



Joonis 13. Administraatori põhifunktsioonid

5 Veebirakenduse analüüs

Veebirakenduste analüüs on iga tarkvara arendusprotsessi lahutamatu osa. See hõlmab rakenduse testimist, testitulemuste ülevaatamist ja vajalike värskenduste tegemist.

Veebirakenduse funktsionaalsuse, kasutusmugavuse ja töökindluse tagamiseks viis autor läbi rea teste. Rakendust testinute seas oli potentsiaalseid kliente, administraatoreid ja juuksurikasutajaid. Nad andsid tagasisidet rakenduse erinevate aspektide kohta.

Nende tagasiside põhjal sai autor analüüsida testitulemusi ja tuvastada valdkonnad, kus saaks teha parendusi.

5.1 Rakenduse testimine

Veebirakenduse testimise etapis osales kokku 12 inimest, neist 5 kliendina ja 2 administraatorina ning 5 juuksuri kasutajana. Testimine viidi läbi nii eemalt kui ka isiklikult ning osalejatel paluti avaldada arvamust rakenduse erinevate aspektide kohta.

Kasutuslihtsuse osas leidis enamik kasutajaid, et äpp on kasutajasõbralik, kuid kujundusega oli probleeme. Mõned kasutajad soovitasid rakenduse atraktiivsemaks muutmiseks disaini täiustada. Kuid keelevaliku funktsiooni hindasid kõik kasutajad, kuna see võimaldas neil rakendust oma eelistatud keeles kasutada.

Funktsionaalsuse osas jäid testijad rakenduse võimalustega rahule. Administraatorikasutajad mainisid ka mõningaid neile saadaolevaid täiustatud funktsioone, näiteks võimalust luua juuksuritele tööaegu, määrata puhkusekuupäevi ja lihtsalt värskendada ettevõtte teavet.

Samuti testiti rakenduse usaldusväärsust ja tulemused olid positiivsed. Kõik kasutajad on leidnud, et rakendus on stabiilne ning vigade ja probleemideta. Mõned administraatorid on siiski väljendanud muret rakenduse turvalisuse pärast ja soovitanud rakendada täiendavaid turvameetmeid, näiteks kaheastmelist autentimist.

Üldiselt jäid testijad rakendusega rahule, kuid mõned punktid vajavad parandamist. On vaja täiustada disaini, et muuta rakendus atraktiivsemaks, samuti rakendada täiendavaid turvameetmeid. Testijad kiitsid aga rakenduse kasutusmugavust, täiustatud funktsioone administraatoritele ja keelevaliku funktsiooni.

5.2 Rakenduse analüüs

Pärast veebirakenduse testide seeriat suutis autor analüüsida tulemusi ja tuvastada parendusvaldkonnad. Potentsiaalsetelt klientidelt, administraatoritelt ja juuksuri kasutajatelt saadud tagasiside aitas authori paremini mõista, kuidas rakendus töötab ja kuidas seda täiustada.

Üks peamisi murekohti testijatele oli rakenduse disain. Kui funktsionaalsust ja kasutusmugavust kiideti, siis disainis peeti puudulikku.

Lisaks leidsid administraatorid, kes rakendust testisid, nende käsutuses olevad erinevad funktsioonid muljetavaldavad. Eriti hinnati oskust genereerida töötajatele töötunde.

Juuksuri kasutajad on leidnud, et rakendus on kasulik, eriti võimalus kontrollida oma ajakava ja kohtumisi. Samuti hindasid nad võimalust kontrollida kohtumisi teiste juuksurite juures, muutes koostöö tegemise ja ajakavade kooskõlastamise lihtsamaks.

Kokkuvõttes võimaldas rakenduse testitulemuste analüüs tuvastada parendusvaldkonnad ja mõista, milliseid uuendusi rakenduses vaja on.

5.3 Rakenduse täiendused

Pärast veebirakenduse testimist tuvastas autor parenduskohad. Rakenduse funktsionaalsuse ja kasutatavuse parandamiseks võidakse aga lisada täiendavaid värskendusi. Rakendust testinud kasutajad on pakkunud välja mõned ideed tulevaste värskenduste jaoks, mida saab rakendada:

- Registreeruge ja logige sisse Apple'i, Facebooki või Google'i kontoga. Apple'i, Facebooki või Google'i kontoga registreerumise ja sisselogimise võimaluse lisamine muudab sisselogimisprotsessi kasutajatele kiiremaks ja mugavamaks.
- Lisage kasutajatele võimalus juuksureid hinnata ja arvustusi jätta, et aidata teistel klientidel teha teadlikke otsuseid.
- Rakendage sisselogimisel turvalisuse suurendamiseks kahefaktorilist autentimist (2FA).
- Lisage rohkem soengufotosid, et anda kasutajatele parem ülevaade sellest, mida nad igalt teenuselt oodata võivad.
- Lisage oma rakendusele rohkem visuaalseid animatsioone. See aitab muuta rakenduse atraktiivsemaks ja lõbusamaks kasutada.
- Lisage oma telefonile või meilile automaatne kohtumise meeldetuletus päev enne kohtumist.

Kokkuvõttes parandavad need väljapakutud uuendused veebirakenduse funktsionaalsust ja kasutatavust. Nende funktsioonide rakendamisel muutub rakendus konkurentsivõimelisemaks ja kasutajasõbralikumaks, meelitades ligi rohkem kasutajaid ja suurendades juurutamispotentsiaali.

6 Kokkuvõte

Selle projekti põhieesmärk oli välja töötada juuksuris aja broneerimiseks veebirakendus, mida saaksid kasutada nii kliendid kui ka juuksuri omanik. Käesolevas töös toodi välja ka projekti esialgsed nõuded, mida oli vaja täita, ja analüüsiti olemasolevaid sarnaseid veebipõhiseid rakendusi kohtumise kokkuleppimiseks, tuues välja nende tugevad ja nõrgad küljed.

Selle eesmärgi saavutamiseks ehitati veebirakenduse tagaosa C# programmeerimiskeelt kasutades. Andmebaas realiseeriti PostgreSQL-i abil, mis sisaldas kogu infot kasutajate, juuksurite, kohtumiste kohta. Esiliidese loomisel kasutati ASP.NET Core MVC raamistikku, pakkudes klientidele ja juuksuritöökodade omanikele reageerivat ja kasutajasõbralikku liidest.

Valmis veebirakendus võimaldab kasutajatel registreerida konto, leppida kokku aeg oma eelistatud kontoga. Juuksurisalongi omanik saab oma äri juhtida, luues ja muutes juuksurigraafikuid, vaadates ja hallates klientide kohtumisi.

Arendusprotsessi käigus testiti rakendust potentsiaalsete klientide, töötajate ja juuksuritöökoja omanikuga, et koguda tagasisidet ja täiustada rakenduse funktsionaalsust. Testitulemusi on analüüsitud ja neid kasutatakse edaspidi veebirakenduse täiustamiseks, muutes selle veelgi mugavamaks ja konkurentsivõimelisemaks.

Üldiselt on juuksuri aja broneerimise veebirakendus edukas projekt, mis vastab esialgsetele nõuetele ja pakub kasulikku teenust nii klientidele, töötajatele kui ka juuksurikoja omanikule. Rakendust saab täiustada, võttes arvesse kasutajate tagasisidet ja lisades uusi funktsioone, mis võivad kasutajakogemust parandada.

Kasutatud kirjandus

[1] „Особенности и основные тенденции развития предприятий индустрии красоты,“ [Võrgumaterjal]. Available: https://cyberleninka.ru/article/n/osobennosti-i-osnovnye-tendentsii-razvitiya-predpriyatiy-industrii-krasoty/viewer. [Kasutatud: 09 märts 2023].

[2] „MVP: что это такое и как работает?,“ (2020, Jun. 30). [Võrgumaterjal]. Available: https://habr.com/ru/companies/productstar/articles/508892/. [Kasutatud: 12 märts 2023].

[3] Webcase, “ ВЕБ ПРИЛОЖЕНИЕ. РАЗНИЦА МЕЖДУ САЙТОМ, ВЕБ-ПРИЛОЖЕНИЕМ, SPA И PWA,” (2022, Feb. 21). [Võrgumaterjal]. Available: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/. [Kasutatud: 13 märts 2023].

[4] Mobilab, „Плюсы и минусы нативных и web приложений,“ [Võrgumaterjal]. Available: https://inlnk.ru/68wPkQ. [Kasutatud: 14 märts 2023].

[5] Positiwise, „Web Application Vs. Desktop Application: Pros and Cons,“ [Võrgumaterjal]. Available: https://positiwise.com/blog/web-application-vs-desktop-application-pros-and-cons/. [Kasutatud: 14 märts 2023].

[6] MyDiv „Web-приложения - преимущества и недостатки,“ [Võrgumaterjal]. Available: https://mydiv.net/arts/view-web-prilozhenija_preimuxhestva_i_nedostatki.html. [Kasutatud: 14 märts 2023].

[7] Aleksandr Makarov, „Что такое веб-приложения, виды и их преимущества,“ (2021, Sep. 22). [Võrgumaterjal]. Available: https://www.azoft.ru/blog/web-apps/. [Kasutatud: 14 märts 2023].

[8] Semen Panichev, „Язык программирования C#: история, специфика, место на рынке,“ (2020, Mar. 11). [Võrgumaterjal]. Available: https://gb.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke. [Kasutatud: 17 märts 2023].

[9] „What is .NET? Introduction and overview,“ (2023, Mar. 15). [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/core/introduction. [Kasutatud: 06 märts 2023].

[10] „Общие сведения ASP.NET Core MVC,“ (2023, Jan. 28). [Võrgumaterjal]. Available: https://learn.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-7.0. [Kasutatud: 17 märts 2023].

[11] Azure, „Что такое PostgreSQL?,“ [Võrgumaterjal]. Available: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-postgresql/. [Kasutatud: 18 märts 2023].

[12] Mark Smallcombe, “ MongoDB vs. PostgreSQL: A Detailed Comparison,“ (2023, Jan. 28). [Võrgumaterjal].Available: https://www.integrate.io/blog/mongodb-vs-postgresql/. [Kasutatud: 19 märts 2023].

[13] Valerie Ulasik, „В Чем Секрет Популярности и Эффективности Методологии Agile,“ (2023, Jan. 20) [Võrgumaterjal]. Available: https://blog.ganttpro.com/ru/metodologiya-agile-methodology/. [Kasutatud: 04 märts 2023].

[14] SkillBox, „Entity Framework: как быстрее написать код для работы с базой данных,“ (2019, Okt. 30) [Võrgumaterjal]. Available: https://skillbox.ru/media/code/entity_framework/. [Kasutatud: 13 märts 2023].

[15] Enlab, „How to build and deploy a three-layer architecture application with C#,“ (2021, Feb. 27) [Võrgumaterjal]. Available: https://enlabsoftware.com/development/how-to-build-and-deploy-a-three-layer-architecture-application-with-c-sharp-net-in-practice.html. [Kasutatud: 07 märts 2023].

[16] SkillBox, „Библиотека jQuery: что это такое и как её подключить,“ (2023, Feb. 14) [Võrgumaterjal]. Available: https://skillbox.ru/media/code/biblioteka-jquery-chto-eto-takoe-i-kak-eye-podklyuchit/. [Kasutatud: 20 märts 2023].

[17] Ella Wilson, „PHP Vs Python: Which Is Best For Web Applications In 2021?,“(2021, Jun. 23) [Võrgumaterjal]. Available: https://medium.com/quick-code/php-vs-python-which-is-best-for-web-applications-in-2021-b7ad3fe0743a. [Kasutatud: 22 märts 2023].

[18] Ella Wilson, „An exceptionally versatile language to build robust apps for the .NET ecosystem,“ [Võrgumaterjal]. Available: https://www.bairesdev.com/technologies/csharp/. [Kasutatud: 23 märts 2023].

[19] Priya Padmkar, „PHP vs C#,“ [Võrgumaterjal]. Available: https://www.educba.com/php-vs-c-sharp/. [Kasutatud: 23 märts 2023].

[20] Peter Wayner, „JavaScript vs. other languages: Live long and prosper?,“ [Võrgumaterjal]. Available: https://techbeacon.com/app-dev-testing/javascript-vs-other-languages-live-long-prosper. [Kasutatud: 23 märts 2023].

[21] Mark Smallcombe, „PostgreSQL vs MySQL: The Critical Differences,“ (2023, Feb. 15). [Võrgumaterjal]. Available: https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/. [Kasutatud: 29 märts 2023].

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

Mina, Mihhail Kuznetsov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose " Veebirakenduse arendamine aja broneerimiseks juuksurisalongis", mille juhendaja on Einar Kivisalu
   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.

15.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti