Web aplikacija za organizaciju rada kozmetičkog salona

Pleško, Tea

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB APLIKACIJA ZA ORGANIZACIJU RADA KOZMETIČKOG SALONA

Završni rad

Tea Pleško

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Tea Pleško
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4706, 28.07.2021.
JMBAG:	0165091805
Mentor:	doc. dr. sc. Krešimir Romić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za organizaciju rada kozmetičkog salona
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	Potrebno je izraditi web aplikaciju koja će omogućiti lakšu organizaciju poslovanja kozmetičkog salona iz perspektive voditelja, ali i korisnika salona. Omogućiti organizaciju zaposlenih osoba po smjenama (tj. vremenskim terminima) u ovisnosti o prostoru kojeg salon ima za različite usluge. Uzeti u obzir da salon može imati više zaposlenika, više prostorija i više usluga koje nudi. Korisnicima salona omogućiti rezervaciju/otkazivanje termina s naglaskom na vizualni prikaz rasporeda dostupnih/nedostupnih termina te prezentaciju usluga koje salon nudi. Tema rezervirana za: Tea Pleško

Datum prijedloga ocjene završnog rada od strane mentora:	18.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Dobar (3)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Dobar (3)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	27.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 27.09.2024.

Ime i prezime Pristupnika:	Tea Pleško
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4706, 28.07.2021.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za organizaciju rada kozmetičkog salona**

izrađen pod vodstvom mentora doc. dr. sc. Krešimir Romić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

U posljednjih nekoliko godina, razvoj tehnologije i interneta implementiran je u gotovo svim aspektima modernog života, a samim time i način na koji poslovni subjekti funkcioniraju. Tradicionalne metode vođenja poslova, poput rukom vođenih kalendara i bilježenje termina, postaju neadekvatne za efikasno upravljanje. Kozmetički saloni suočavaju se s nizom takvih izazova u svakodnevnom poslovanju. Upravljanje terminima klijenata, koordinacija zaposlenih i vođenje evidencije o pruženim uslugama samo su neki od zadataka koji su kompleksni i ono najvažnije, vremenski zahtjevni. Tradicionalne metode vođenja ovih procesa često dovode do raznih grešaka, od kojih su neke dvostruke rezervacije, preklapanje termina, propušteni termini i sl. Ovakvo vođenje može uzrokovati velikim nezadovoljstvom klijenata, a samim time i negativnim utjecajem na reputaciju i profitabilnost salona. Implementacija web aplikacije za organizaciju rada kozmetičkog salona predstavlja neophodan korak ka unapređenju, modernizaciji i olakšanju poslovnih procesa te poboljšanju korisničkog iskustva.

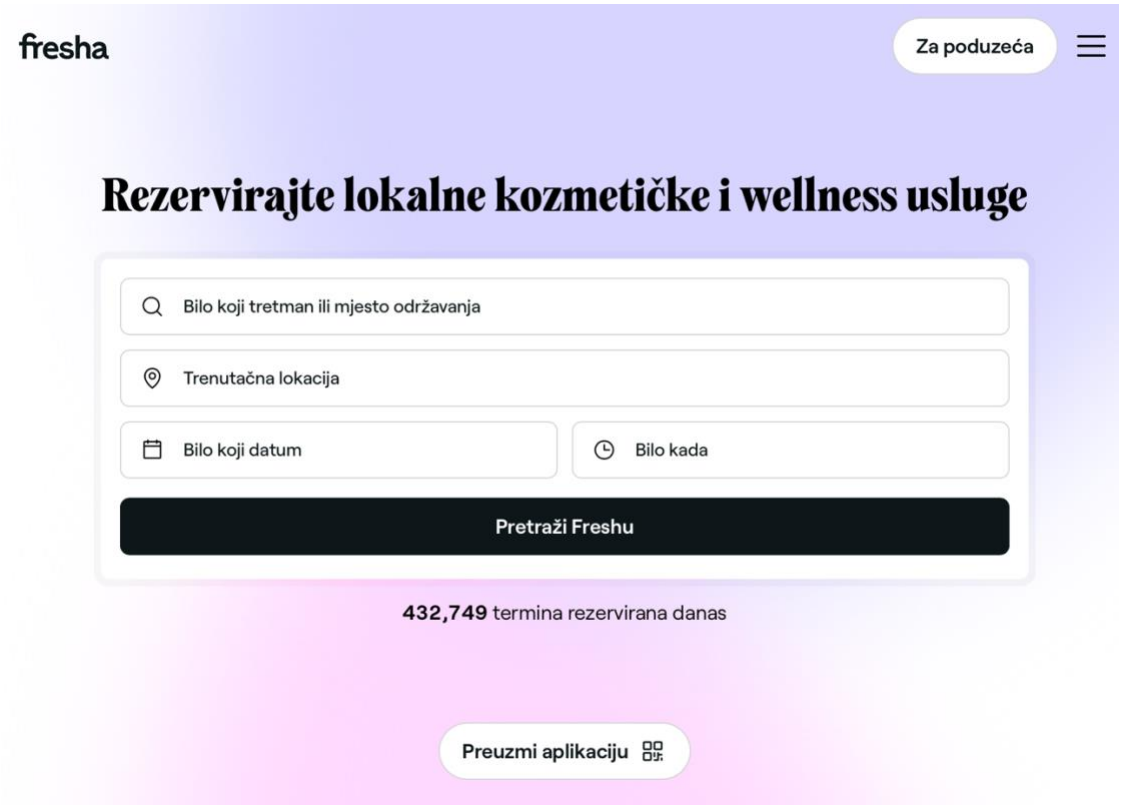
1.1. Zadatak završnog rada

Potrebno je izraditi web aplikaciju koja će omogućiti lakšu organizaciju poslovanja kozmetičkog salona iz perspektive voditelja, ali i korisnika salona. Omogućiti organizaciju zaposlenih osoba po smjenama (tj. vremenskim terminima) u ovisnosti o prostoru kojeg salon ima za različite usluge. Uzeti u obzir da salon može imati više zaposlenika, više prostorija i više usluga koje nudi. Korisnicima salona omogućiti rezervaciju/otkazivanje termina s naglaskom na vizualni prikaz rasporeda dostupnih/nedostupnih termina te prezentaciju usluga koje salon nudi.

2. SLIČNE APLIKACIJE

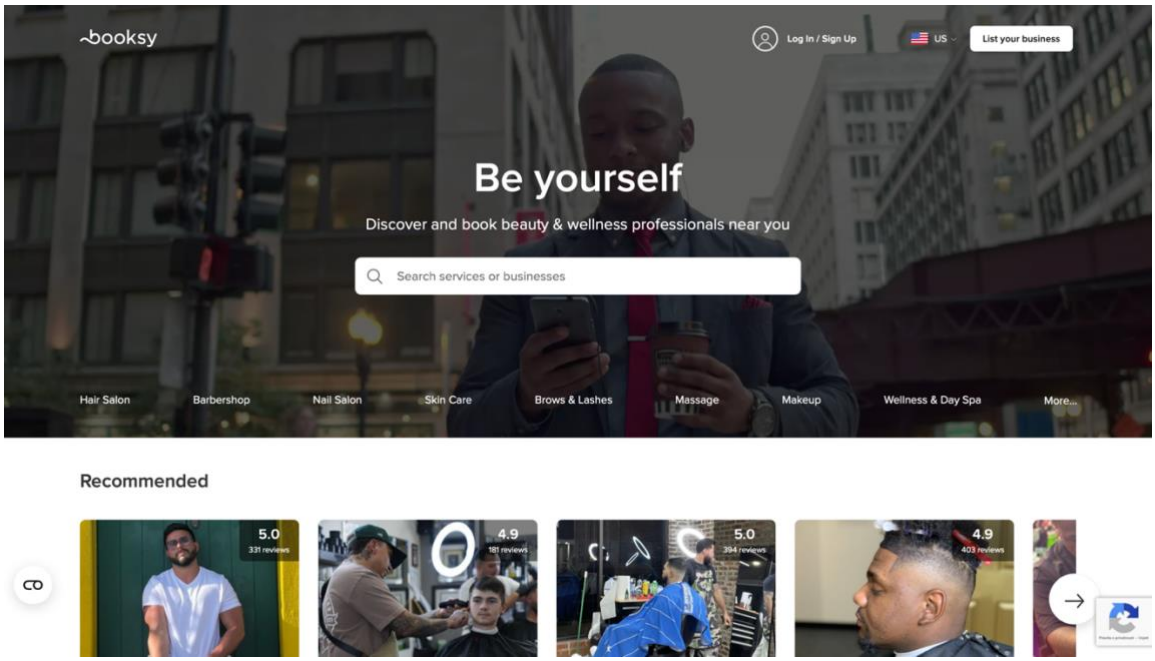
Tržište je bogato rješenjima koja nude širok spektar funkcionalnosti za organizaciju rada kozmetičkih salona. Međutim, svaki od ovih sistema ima svoje prednosti i nedostatke. Pojedini saloni nude vlastitu web aplikaciju koja je stvorena na način da funkcionira samo za vlastite potrebe. Primjer takve web aplikacije bit će aplikacija ovog završnog rada.

Jedna od popularnih aplikacija za upravljanje terminima i poslovanjem u kozmetičkim salonima je Fresha (Slika 2.1.). Aplikacija nudi brojne funkcionalnosti poput online zakazivanja termina, upravljanja kalendarom zaposlenih, automatskih podsjetnika za klijente putem SMS-a i email-a, upravljanje inventarom i prodajom proizvoda. Prednost ove aplikacije je to što je osnovna verzija sa širokim spektrom funkcionalnosti besplatna te su integrirani različiti platni sistemi. Kada je riječ o nedostacima, jasno je da su postojanjem osnovne besplatne verzije, naprednije funkcionalnosti dostupne samo u plaćenim verzijama. Još jedan nedostatak je taj što je ograničena prilagodljivost specifičnim potrebama pojedinačnih salona.



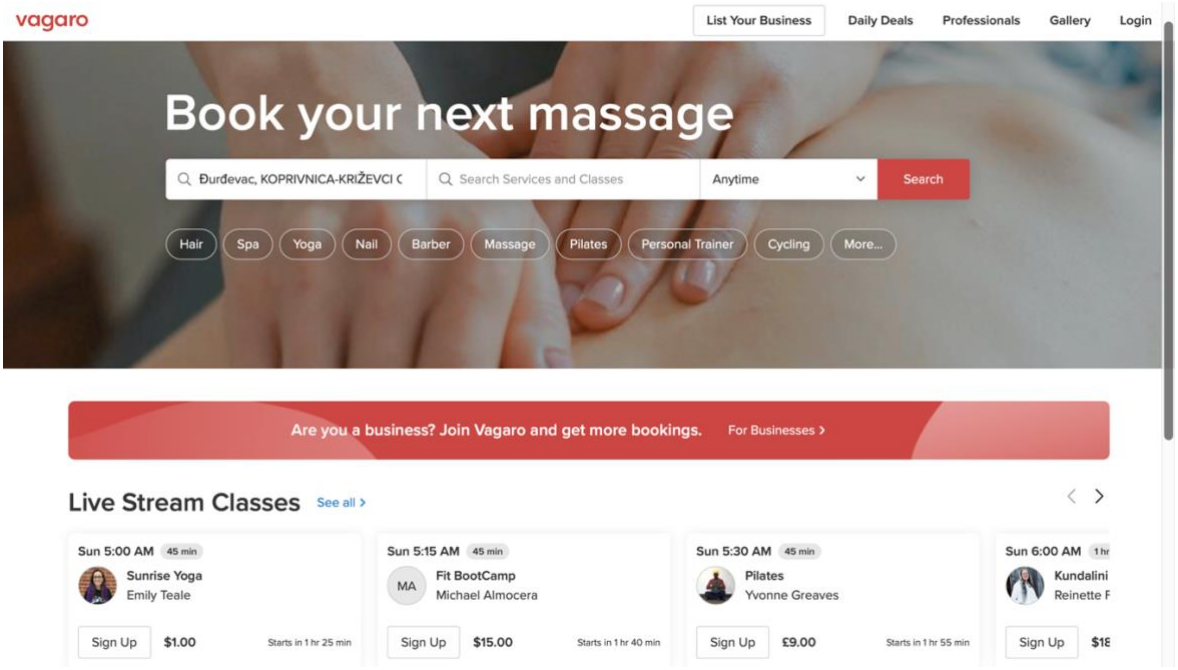
Slika 2.1. Prikaz početne stranice aplikacije Fresha

Booksy (Slika 2.2.) je popularna aplikacija za upravljanje termina koja uključuje online rezervacije, praćenje klijenata i marketing. Klijentima omogućuje rezervacije putem web aplikacije, automatizirane podsjetnike, posebne ponude i recenzije te je sinkronizirana s kalendarima, društvenim mrežama i financijskim sustavima. Nedostaci su ti što mjesečna naknada za manje salone može biti visoka i što može biti složena za početnike zbog brojnih opcija i postavki.



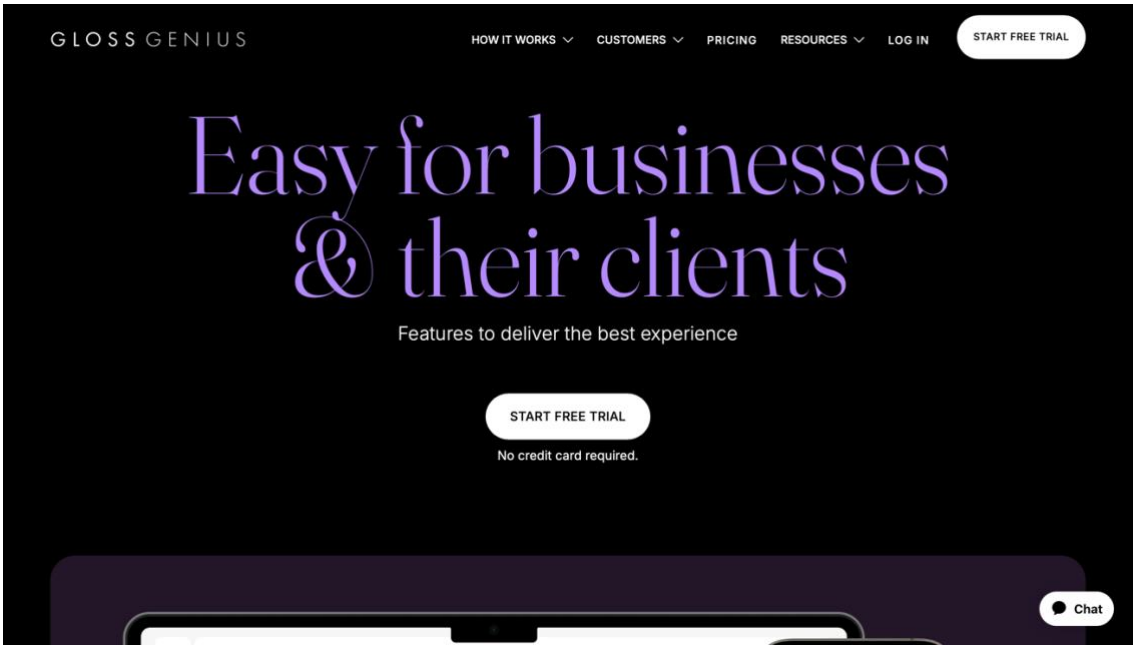
Slika 2.2. Prikaz početne stranice aplikacije Booksy

Vagaro (Slika 2.3.), svestrana aplikacija koja nudi upravljanje zalihama, prodaju proizvoda, izvještavanje, online rezervacije i marketing, omogućuje interakciju sa salonom putem postojećih web stranica ali i jednostavan pristup rasporedima i klijentima putem mobilne aplikacije. Saloni se mogu oglašavati putem Vagaro platforme, privlačeći nove klijente.



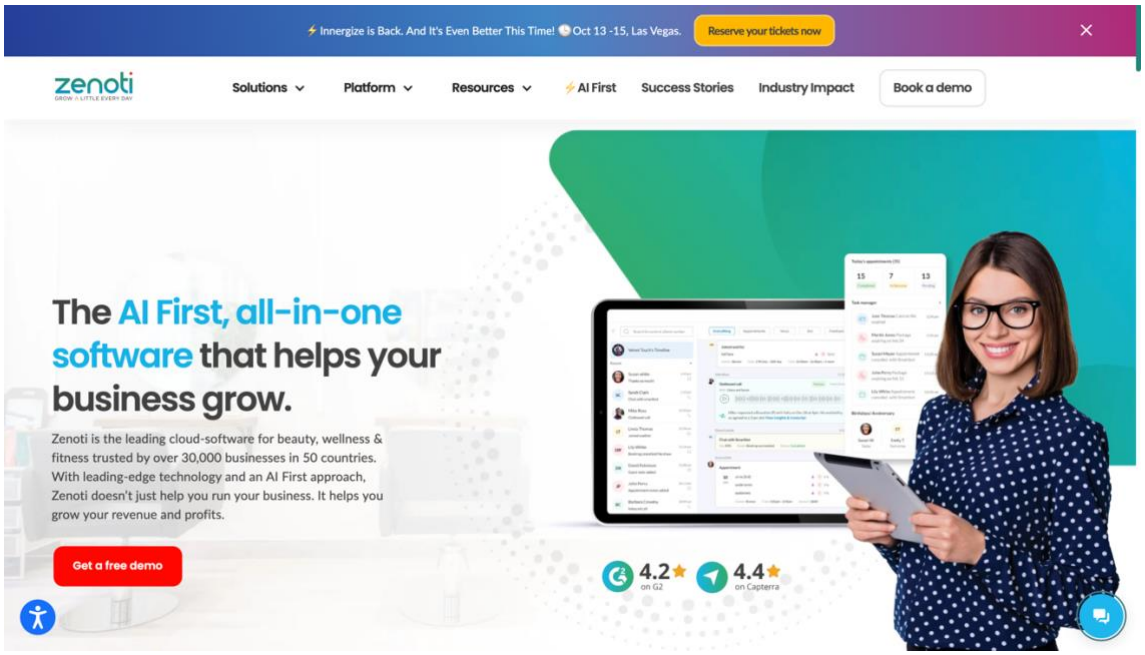
Slika 2.3. Prikaz početne stranice aplikacije Vagaro

GlossGenius (Slika 2.4.) je intuitivna i vizualno privlačna aplikacija koja omogućuje klijentima jednostavno rezerviranje, uz ugrađeni POS sustav plaćanje online te automatske podsjetnike i izvještavanja. Aplikacija nije besplatna, a mjesečna naknada može biti veća za mala poduzeća.



Slika 2.4. Prikaz početne stranice aplikacije GlossGenius

Zenoti (Slika 2.5.) je napredna platforma za upravljanje kozmetičkim salonima, spa centrima i fitness studijima, s naglaskom na velike salone. Složenija je za implementaciju i zahtijeva vrijeme za obuku zaposlenika. Zbog toga što je prikladnija za velike salone i lance, manji saloni možda neće koristiti sve mogućnosti aplikacije, čineći je manje isplativom.



Slika 2.5. Prikaz početne stranice aplikacije Zenoti

3. ARHITEKTURA APLIKACIJE

U kontekstu izrade web aplikacije za kozmetički salon, arhitektura aplikacije igra ključnu ulogu u osiguravanju da rješenje bude kompaktno i jednostavno za korištenje. Ova aplikacija osmišljena je kako bi zadovoljila specifične potrebe salona, uključujući upravljanje rezervacijama, praćenje klijenata, te pružanje raznih kozmetičkih usluga. U ovom poglavlju razmotrit ćemo korištene tehnologije i alate prilikom izrade aplikacije.

3.1. Korištene tehnologije i alati

3.1.1. HTML

HTML (engl. *HyperText Markup Language*) je kod koji se koristi za strukturiranje web stranice i njezinog sadržaja. Na primjer, sadržaj može biti strukturiran unutar skupa odlomaka, popisa točaka s grafičkim oznakama ili pomoću slika i podatkovnih tablica. HTML je jezik koji koristi oznake i definira strukturu sadržaja. HTML se sastoji od niza elemenata koje koristite za "omotavanje" različitih dijelova sadržaja kako bi izgledao ili djelovao na određeni način. Oznake koje ga okružuju mogu napraviti hipervezu riječi ili slike na neko drugo mjesto, mogu staviti riječi u kurziv, mogu povećati ili smanjiti font i tako dalje [1].

Svaki HTML dokument sastoji se od zaglavlja i tijela. U zaglavlju dokumenta definiramo elemente koji se ne prikazuju na stranici. Između ostalog, ovdje definiramo naslov dokumenta koji se prikazuje na naslovima kartica web preglednika. Sve ostalo, sav sadržaj koji se prikazuje na stranici, smještamo u tijelo dokumenta. Namjena web preglednika (Chrome, Edge, Safari, itd.) je pročitati HTML dokument i ispravno ga prikazati. Preglednik ne prikazuje HTML tagove, ali koristi ih kako bi odlučio kako prikazati dokument (primjer <h1> element definira veliki naslov, <p> paragraf i slično) [2].

3.1.2. CSS

HTML nikada nije bio zamišljen da sadrži oznake za oblikovanje web stranice. Kada su oznake poput <font> i atributi za boju dodani u HTML 3.2 specifikaciju, to je stvorilo noćnu moru za web programere. Razvoj velikih web stranica, gdje su informacije o fontovima i bojama morale biti dodane na svaku pojedinu stranicu, postao je dug i skup proces. Kako bi se riješio taj problem, World Wide Web Consortium (W3C) stvorio je CSS (engl. *Cascading Style Sheets*) [3]. CSS je jezik čija je uloga da pojednostavi proces dizajniranja i omogući da web stranica postane prezentirajuća izgledom. Omogućuje primjenu stilova na HTML dokumente, opisujući kako bi web stranica trebala izgledati

propisivanjem boja, fontova, razmaka i pozicioniranja. CSS je jednostavan za naučiti i razumjeti te programerima i dizajnerima pruža moćnu kontrolu nad prezentacijom HTML elemenata. HTML koristi oznake, a CSS skup pravila. CSS stilovi se primjenjuju na HTML element pomoću selektora [4].

3.1.3. JavaScript

JavaScript je skriptni jezik kojim možemo kreirati dinamičke web stranice. Dinamičke web stranice omogućavaju međudjelovanje s korisnikom, upravljanje web preglednikom ili dinamičko kreiranje web stranice. Izvodi se na klijentskom računalu i može se izvoditi u svim modernim web preglednicima. ECMA-262 je službeni naziv JavaScript standarda, a trenutna inačica 5.1 donesena je 2011. godine [5]. JavaScript nije pojednostavljena inačica programskog jezika Java. Povezuje ih jedino slična sintaksa i to što se koriste za izvršavanje određenih radnji unutar preglednika. Izvorno se JavaScript trebao zvati LiveScript, ali da bi se potakla uporaba novog skriptnog jezika, nazvan je slično jeziku Java, od kojeg se u tadašnje vrijeme dosta očekivalo [6].

Neke od prednosti JavaScripta su:

- manja potreba za komunikaciju sa serverom (može se provjeriti ispravnost podatka prije slanja stranice na server što smanjuje promet prema serveru)
- korisnik ne treba čekati da se stranica ponovno učita kako bi vidio je li zaboravio unijeti neki podatak
- moguće je kreirati sučelje kojim korisnik može upravljati pritiskom na miš ili tipkovnicu čime se povećava interaktivnost
- moguće je uhvatiti ili pustiti komponente te animirati bogato sučelje prema korisniku

3.1.4. React

React je JavaScript biblioteka za izradu korisničkog sučelja, koja je nastala 2013. godine u Facebooku. Glavni razvojni tim činili su Jordan Walke i Andrew Clark, koji su željeli riješiti problem kompleksnosti izrade sučelja za web aplikacije. React je zasnovan na ideji komponenti i omogućuje izradu dinamičnog korisničkog sučelja bez potrebe za cjelovitom obnovom cijele stranice. To dovodi do veće brzine i performansi, kao i lakšeg održavanja koda. Danas se koristi u velikom broju web aplikacija, a njegova popularnost i dalje raste. To je zbog jednostavnosti i efikasnosti koju pruža, kao i zbog toga što se lako integrira s drugim tehnologijama, poput Node.js i Redux. React se također

koristi u izradi mobilnih aplikacija, korištenjem React Native, tehnologije koja omogućuje izradu mobilnih aplikacija na istom kodu kao i web aplikacije [7].

Popularnost Reacta danas je zasjenila sve ostale *frontend* razvojne okvire. Evo zašto:

- olakšava izradu dinamičkih web aplikacija jer zahtijeva manje kodiranja i nudi više funkcionalnosti, za razliku od JavaScripta, gdje kodiranje često postane vrlo složeno
- koristi Virtual DOM, što omogućuje bržu izradu web aplikacija (virtual DOM uspoređuje prethodna stanja komponenti i ažurira samo one stavke u stvarnom DOM-u koje su promijenjene, umjesto da ponovno ažurira sve komponente)
- komponente su osnovni gradivni blokovi svake React aplikacije, a jedna aplikacija obično se sastoji od više komponenti (komponente imaju svoju logiku i kontrole, te se mogu ponovno koristiti kroz cijelu aplikaciju, što značajno smanjuje vrijeme potrebno za razvoj aplikacije)
- prati jednosmjerni protok podataka što znači da prilikom dizajniranja React aplikacije, programeri često ugrađuju child komponente unutar parent komponenti [8]

3.1.5. Firebase

Firebase je razvojna platforma za mobilne i web-aplikacije. Pomoću Firebasea je moguće brzo izraditi aplikacije visoke kvalitete, povećati bazu angažiranih korisnika i ostvariti veću zaradu. Platforma uključuje nekoliko dobro integriranih značajki koje možete kombinirati, uključujući analitiku, pozadinske programe za mobilne uređaje te alate za razvoj i unovčavanje aplikacija za maksimalno povećanje uspjeha aplikacije [5].

Ključne značajke Firebase-a:

- Realtime Database i Firestore (baze podataka koje omogućuju pohranu i sinkronizaciju podataka u stvarnom vremenu)
- Authentication (pruža jednostavan način za autentifikaciju korisnika putem različitih metoda, uključujući e-mail, lozinke, Google prijavu i druge)
- Cloud Functions (omogućuje pisanje i izvršavanje serverless funkcija u odgovoru na događaje iz Firebase-a i drugih izvora)
- Cloud Messaging (omogućuje slanje push obavijesti korisnicima aplikacije)
- Analytics (pruža detaljne uvide u ponašanje korisnika i performanse aplikacije)
- Crashlytics (alat za praćenje i upravljanje greškama u aplikaciji)

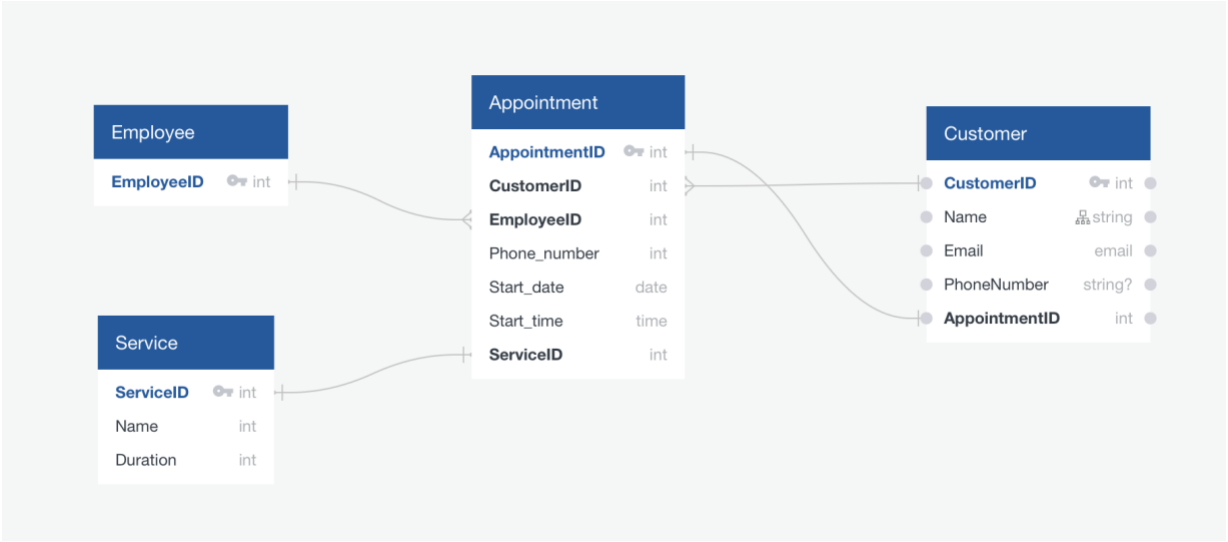
- Hosting (pruža jednostavan način za hostanje web aplikacija s globalnom mrežom dostave sadržaja (CDN))

3.2. Arhitektura baze podataka

Arhitektura baze podataka služi kao osnova za upravljanje i organizaciju podataka u *backendu* aplikacije. U modelu baze podataka na slici 3.1. prikazano je nekoliko povezanih tablica koje omogućavaju efikasno praćenje informacija o zaposlenicima, korisnicima, terminima i uslugama. Ovaj model koristi ključne atribute i relacije kako bi omogućio jednostavno skladištenje i pristup podacima.

Tablica Employee sadrži podatke o zaposlenicima koji pružaju usluge korisnicima. Svaki zaposlenik je identificiran jedinstvenim identifikatorom EmployeeID, koji predstavlja primarni ključ ove tablice. Ovaj identifikator koristi se u drugim tablicama, prvenstveno u tablici Appointment, kako bi se moglo precizno odrediti koji je zaposlenik zadužen za određeni termin. Zaposlenici su ključni element ove baze podataka jer su oni odgovorni za pružanje usluga i izravno komuniciraju s korisnicima. Ova tablica omogućava pohranjivanje osnovnih informacija o zaposlenicima, kao što su osobni podaci, te je povezana s ostalim dijelovima baze putem stranih ključeva. Na taj način, sustav može brzo i precizno povezati zaposlenike s rezerviranim terminima.

Tablica Appointment je središnji element ovog modela baze podataka, jer pohranjuje informacije o svim rezerviranim terminima. Svaki termin identificiran je jedinstvenim



Slika 3.1. Tablice relacija baze podataka

ključem AppointmentID, koji omogućuje jednoznačno razlikovanje između pojedinih termina. Ova tablica sadrži nekoliko važnih atributa koji omogućuju praćenje svih bitnih podataka vezanih uz svaki pojedini termin. Prvi od tih atributa je CustomerID, koji predstavlja strani ključ i povezuje tablicu Appointment s tablicom Customer. Ova veza omogućava praćenje informacija o korisniku koji je rezervirao termin. Sljedeći ključni atribut je EmployeeID, koji predstavlja strani ključ povezan s tablicom Employee. Ova relacija omogućava povezivanje svakog termina s odgovarajućim zaposlenikom koji će pružiti uslugu. Još jedan važan atribut ove tablice je Phone_number, koji sadrži telefonski broj korisnika. Ovaj podatak olakšava komunikaciju između zaposlenika i korisnika prije samog termina. Zatim, tu su atributi Start_date i Start_time, koji označavaju datum i vrijeme početka termina, čime se omogućava pravilno planiranje i raspoređivanje vremena zaposlenika. Na kraju, atribut ServiceID predstavlja strani ključ koji povezuje termin s određenom uslugom iz tablice Service. Tablica Appointment ima ključnu ulogu u ovoj bazi podataka jer omogućava povezivanje svih ključnih informacija – korisnika, zaposlenika, usluga, datuma i vremena – u jedinstvenu cjelinu koja olakšava vođenje evidencije o zakazanim terminima.

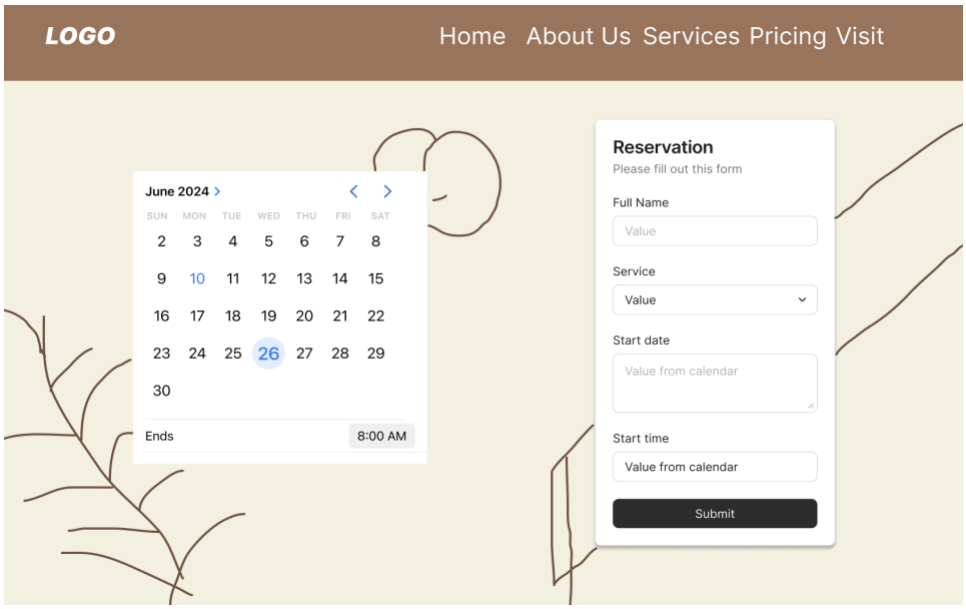
Tablica Customer sadrži informacije o korisnicima koji koriste usluge. Svaki korisnik ima svoj jedinstveni identifikator, CustomerID, koji predstavlja primarni ključ ove tablice. Osnovni podaci koji se pohranjuju u ovoj tablici uključuju ime korisnika (Name), njegovu email adresu (Email) te broj telefona (PhoneNumber). Osim osnovnih osobnih podataka, tablica Customer povezana je s tablicom Appointment putem stranog ključa AppointmentID. Ova relacija omogućava praćenje termina koje je određeni korisnik rezervirao. Time se olakšava praćenje povijesti usluga koje je svaki korisnik koristio, a zaposlenici mogu lako doći do podataka o kontaktu korisnika za eventualne promjene ili potvrde termina. Tablica Customer je od velike važnosti jer omogućava sustavu da pohranjuje sve relevantne informacije o korisnicima na centraliziran način, a povezanost s tablicom termina olakšava upravljanje podacima o rezervacijama.

Tablica Service pohranjuje podatke o uslugama koje su dostupne korisnicima. Svaka usluga identificirana je primarnim ključem ServiceID. Glavni atributi ove tablice su Name, koji predstavlja naziv usluge, i Duration, koji označava trajanje svake pojedine usluge. Ova tablica povezana je s tablicom Appointment putem stranog ključa ServiceID, čime se osigurava povezivanje svake rezervacije termina s odgovarajućom uslugom. To omogućuje sustavu da precizno prati koje su usluge rezervirane i u kojem trajanju, što je važno za raspoređivanje zaposlenika i optimalno korištenje

resursa. Tablica Service omogućuje fleksibilno upravljanje ponudom usluga unutar sustava, a povezanost s ostalim tablicama osigurava da se usluge pravilno vežu uz termine i zaposlenike.

3.3. Dizajn korisničkog sučelja

Korisničko sučelje (engl. *User interface - UI*) jedan je od najvažnijih aspekata svake aplikacije jer direktno utječe na korisničko iskustvo (engl. *User Experience - UX*). Cilj dizajniranja korisničkog sučelja ovog rada bio je osigurati intuitivan, pregledan i estetski privlačan dizajn sučelja, prilagođen potrebama korisnika koji žele jednostavno rezervirati termine za različite kozmetičke usluge. Kombinacija boja, poput bež i smeđe, te drugih neutralnih tonova odražava uglađeni i luksuzni brend, a svi elementi sučelja, uključujući kalendare, forme i gumbe, harmonično se uklapaju u vizualnu temu aplikacije. Jedna od temeljnih smjernica dizajna bila je jednostavnost, s naglaskom na to da aplikacija ostane intuitivna za korisnike svih dobnih skupina. Kako korisnici u aplikaciji ne trebaju kreirati profile ili se prijavljivati, dizajn je usmjeren na jednostavnost interakcije bez komplikacija. Drugim riječima, naglasak je stavljen na kalendar prema kojem korisnici mogu vidjeti zauzete i slobodne termine te jednostavno kreirati svoju rezervaciju unosom nekih osnovnih osobnih podataka. Prikaz osnovne ideje izgleda ispunjavanja rezervacije i odabira datuma vidljiv je na slici 3.2. Dizajn aplikacije napravljen je u Figmi, programu za dizajniranje prilikom razvoja web i drugih aplikacija. No, imajući na umu jednostavnost dizajna aplikacije, nije bilo previše mjesta u dizajniranju aplikacije prije same izrade stoga dizajn ne pokriva svaku sekciju aplikacije. Većina dizajnerskog dijela je odrađena u hodu, poput izbora pozadinskih fotografija, dodatno prilagođavanje nijansi boja i slično.



Slika 3.2. Primjer dizajna kreiranja rezervacije u Figmi

4. IMPLEMENTACIJA

Ovo poglavlje obuhvaća implementaciju zahtjeva web aplikacije kozmetičkog salona uz pojašnjenje i prikaz samog koda i logike kojom se dolazilo do rješenja.

4.1. Analiza i objašnjenje ključnih dijelova HTML koda

Dokument naziva indeks.html predstavlja strukturu aplikacije salona ljepote. Aplikacija omogućuje korisnicima pregled informacija o salonu, uslugama, cijenama, te rezervaciju termina putem integriranog kalendara. Unutar <head> sekcije indeks.html-a nalaze se meta tagovi koji su dio svakog HTML dokumenta, link za vanjsku biblioteku ikona (koristi se za prikaz ikona unutar aplikacije, Font Awesome), uključivanje biblioteke za manipulaciju datumima i vremenom, što je ključno za kalendar i rezervacije te putanje prema dvjema glavnim CSS datotekama od kojih se jedna koristi za stiliziranje cijele stranice, a druga specifično stilizira formu za rezervaciju termina i prikaz kalendara.

4.1.1. Zaglavlje i navigacija

Zaglavlje i navigacija predstavljaju ključne dijelove svake web stranice jer omogućuju korisnicima jednostavan pristup različitim sekcijama stranice, kao i ključnim informacijama kao što su naziv brenda i meni za navigaciju. U ovom slučaju, zaglavlje (Slika 4.1.) i navigacija služe kao korisnički vodič za pregled i korištenje stranice salona. Funkcija Logo služi kao vizualni identitet stranice. Koristi se `<a>` tag s atributom `href="#"` što znači da klik na logo vodi korisnika na početnu stranicu ili ga zadržava na vrhu iste. Unutar `<nav>` taga s klasom navbar definiran je glavni navigacijski blok koji sadrži veze (linkove) prema različitim sekcijama na stranici. Svaki `<a>` predstavlja jedan navigacijski link koji omogućuje korisnicima da brzo prelaze iz jedne sekcije u drugu bez potrebe za skrolanjem. Npr. `href="#home"` vodi do početne sekcije web stranice, `href="#about"` vodi do sekcije „About Us“, `href="#services"` vodi do sekcije usluga, itd. Element menu-btn služi kao gumb koji otvara i zatvara navigacijski meni na mobilnim uređajima. Ikona s klasom fa-bars koristi se iz vanjske biblioteke Font Awesome i prikazuje tri horizontalne linije, što je prepoznatljiv simbol za meni.

```
15    <section class="header">
16
17        <a href="#" class="logo"> BeYoutiful </a>
18
19        <nav class="navbar">
20            <div id="close-navbar" class="fas fa-times"></div>
21            <a href="#home">home</a>
22            <a href="#about">about</a>
23            <a href="#services">services</a>
24            <a href="#pricing">pricing</a>
25            <a href="#visit-main">book a visit</a>
26            <a href="#review">review</a>
27            <a href="#" id="admin-link">Admin</a>
28        </nav>
29
30        <div id="menu-btn" class="fas fa-bars"></div>
31
32    </section>
```

Slika 4.1. Kod zaglavlja i navigacije

4.1.2. Sekcije

Svaka sekcija ima specifičnu svrhu i sadrži informacije povezane s određenim dijelovima stranice. Sekcije u ovoj aplikaciji uključuju početnu stranicu (home), O nama (engl. *about*), Usluge (engl. *services*), Cjenik (engl. *pricing*), Rezervacije(engl. *visit-main*) i Recenzije (engl. *review*). Svaka sekcija ima svoj jedinstveni identifikator (id) koji omogućava korisnicima brzi pristup različitim dijelovima stranice putem navigacije. Sekcije Rezervacije i Recenzije obrađene su u zasebnom poglavlju.

Sekcija „Home“ (Slika 4.2.) služi kao uvodni dio stranice i ovdje se nalazi poruka brenda koja poziva korisnike da otkrije usluge salona ljepote. Sekcija O nama (engl. *About Us*), čiji kod je prikazan na slici 4.3., informira posjetitelje o timu i vrijednostima salona. Naslov je označen `<h1>` tagom kako bi istaknuo važnost te sekcije. Dodana je i slika te opis unutar `<p>` paragrafskog teksta koji detaljnije govori o timu i filozofiji rada. Na kraju, `<div class="icons-container">` grafički prikazuje specifične usluge putem ikona i atribute salona uz prateći opis. Sekcija Usluge (engl. *Services*) informativno prikazuje ključne usluge koje salon nudi zajedno sa slikama i kratkim opisom. Usluge su predstavljene kroz više „box“ elemenata (Slika 4.4.) koji sadrže sliku usluge i njezin naziv. Funkcija sekcije Cjenik (Slika 4.5.) je prikazati ponude i cijene ključnih usluga salona. Cjenik je

prikazan kroz `<div class="box-container">` čiji elementi sadrže ime usluge, cijenu i listu specifičnih tretmana.

```
37    <section class="home" id="home">
38
39        <div class="content">
40            <span>Welcome</span>
41            <h3>Unveil Your Radiance:  <br> Beauty, Elegance, <br> You.</h3>
42        </div>
43
44    </section>
```

Slika 4.2. Kod sekcije „Home“

```
49  <section class="about" id="about">
50
51      <h1 class="heading">About us</h1>
52
53      <div class="row" >
54
55          <div class="image">
56              <img src="images/about.jpg" alt="">
57          </div>
58
59          <div class="content">
60              <h3 class="title">We are a group of passionate beauty professionals</h3>
61              <p>With a passion for excellence, our team of highly skilled professionals is dedicated to staying at the forefront of
62                  beauty trends and techniques. We pride ourselves on having undergone extensive training, both locally and internationally.
63                  Our experts have participated in numerous educational programs and workshops, ensuring that we are equipped with the latest
64                  knowledge and skills to offer you the best in beauty treatments.</p>
65              <p>At BeYoutiful, we are more than just a salon; we are your partners in beauty and well-being. Join us and experience the
66                  difference that expertise, care, and passion can make.</p>
67
68              <div class="icons-container">
69                  <div class="icons">
70                      <img src="images/manicure.png" alt="">
71                      <h3>Professional tools</h3>
72                  </div>
73                  <div class="icons">
74                      <img src="images/beauty.png" alt="">
75                      <h3>quality products</h3>
76                  </div>
77                  <div class="icons">
78                      <img src="images/makeup.png" alt="">
79                      <h3>Nails, makeup and more</h3>
80                  </div>
81              </div>
82          </div>
83      </div>
84  </section>
```

Slika 4.3. Kod sekcije „O nama“

```
<section class="services" id="services">

    <h1 class="heading">premium services</h1>

    <div class="box-container">

        <div class="box">
            <img src="images/services-1.jpg" alt="">
            <div class="content">
                <h3>Makeup</h3>
            </div>
        </div>

        <div class="box">
            <img src="images/services-2.jpg" alt="">
            <div class="content">
                <h3>Nails </h3>
            </div>
        </div>

        <div class="box">
            <img src="images/services-3.jpg" alt="">
            <div class="content">
                <h3>Depilation</h3>
            </div>
        </div>

        <div class="box">
            <img src="images/services-4.jpg" alt="">
            <div class="content">
                <h3>Solarium</h3>
            </div>
        </div>
    </div>
</section>
```

Slika 4.4. Kod sekcije „Usluge“

```
<section class="pricing" id="pricing">

    <h1 class="heading">offer and Pricing</h1>

    <div class="box-container">

        <div class="box">
          <h3 class="title">Nails</h3>
          <div class="price">
            <span class="currency">€</span>
            <span class="amount">40</span>
          </div>
          <ul>
            <li> <i class="fas fa-check"></i> Regular nail polis </li>
            <li> <i class="fas fa-check"></i> Gel nails </li>
            <li> <i class="fas fa-check"></i> Acrylic nails </li>
            <li> <i class="fas fa-check"></i> Nail art </li>
          </ul>
        </div>

        <div class="box">
          <h3 class="title">Depilation</h3>
          <div class="price">
            <span class="currency">€</span>
            <span class="amount">30</span>
          </div>
          <ul>
            <li> <i class="fas fa-check"></i> Waxing</li>
            <li> <i class="fas fa-check"></i> Sugaring </li>
            <li> <i class="fas fa-check"></i> IPL</li>
            <li> <i class="fas fa-check"></i> Threading</li>
          </ul>
```

Slika 4.5. Kod sekcije „Cjenik“

4.1.3. Podnožje

HTML kod na slici 4.6. predstavlja podnožje web stranice (engl. *footer*) s osnovnim informacijama i povezanim funkcionalnostima putem JavaScript skripti. Podnožje se sastoji od tri glavne sekcije. Prva sekcija sadrži ikone za društvene mreže (Facebook, Twitter, Instagram, i LinkedIn), gdje su ikone implementirane pomoću font-awesome biblioteke, a namijenjene su za dijeljenje ili povezivanje s društvenim profilima. Druga sekcija sadrži kontakt informacije, uključujući telefonski broj i poveznicu na e-mail adresu. Treća sekcija prikazuje lokaciju, s fiktivnim primjerom adrese. Uz HTML, na kraju dokumenta (Slika 4.7.) učitane su različite JavaScript biblioteke i skripte koje omogućuju dodatne funkcionalnosti stranice. Među njima su Firebase za *backend* usluge, Moment.js za rad s datumima, te skripte poput script.js i reviews.js koje upravljaju specifičnim funkcijama, uključujući administraciju i rukovanje recenzijama. Ove skripte omogućuju interaktivnost i dinamički sadržaj na stranici.

```
<section class="footer">
    <div class="box-container">
        <div class="box">
            <h3> Find us here </h3>
            <div class="share">
                <a href="#" class="fab fa-facebook-f"></a>
                <a href="#" class="fab fa-twitter"></a>
                <a href="#" class="fab fa-instagram"></a>
                <a href="#" class="fab fa-linkedin"></a>
            </div>
        </div>
        <div class="box">
            <h3>contact us</h3>
            <p>+385 123 456 789</p>
            <a href="#" class="link">tea.plesko15@gmail.com</a>
        </div>
        <div class="box">
            <h3>localization</h3>
            <p> Street 1 <br>
              City <br>
              Country</p>
        </div>
    </div>
    <div class="credit"> created by <span>teaplesko</span> | all rights reserved! </div>
</section>
```

Slika 4.6. Kod podnožja HTML-a

```
276   <script src="https://cdnjs.cloudflare.com/ajax/libs/firebase/7.14.1-0/firebase.js"></script>
277   <script src="js/script.js"></script>
278   <script src="dist/main-bundle.js"></script>
279   <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js"></script>
280   <script type="module" src="js/admin-login.js"></script>
281   <script src="js/reviews.js"></script>
```

Slika 4.7. Učitavanje biblioteka i skripti na kraju HTML-a

4.2. Rezerviranje termina

Ranije spomenuta sekcija za rezervaciju termina u korijenskom HTML-u samo poziva i predviđa mjesto gdje će se generirati i prikazati kalendar te forma koju je potrebno ispuniti ako korisnik želi rezervirati termin. Kod koji je smješten u korijenski HTML prikazan je slikom 4.8. Kalendarom i obrascem upravlja se dinamički preko drugih skripti.

```
187   <section class="visit-main" id="visit-main">
188       <h1 class="heading"> book a visit </h1>
189
190       <div class="row">
191           <div class="calendar-form-container" id="calendar-form-container">
192           </div>
193       </div>
194   </section>
```

Slika 4.8. Kod sekcije „Visit“

4.2.1. Kalendar koji prati rezervirane/slobodne termine prema zaposlenicima

Ključna stvar koja čini ovu web aplikaciju funkcionalnom jest upravo kalendar (engl. *Scheduler*). Komponenta *Scheduler* omogućava korisnicima odabir datuma i vremena za rezervaciju, a podaci se spremaju u Firebase bazu podataka. Kod kojim je implementirana funkcionalnost za zakazivanje termina dobiven je koristeći React, Firebase i komponentu kalendara iz biblioteke react-big-calendar. Iz tog razloga je najprije bilo potrebno uvesti (Slika 4.9.) ključne React funkcionalnosti (useState, useEffect), kalendar (react-big-calendar), moment.js za rad s datumima, Firebase module za interakciju s bazom podataka, te CSS datoteku za stiliziranje forme. Sljedeća ključna stvar jest konfiguracija i inicijalizacija Firebase-a (Slika 4.10.).

```
1  import React, { useState, useEffect } from 'react';
2  import { Calendar, momentLocalizer } from 'react-big-calendar';
3  import moment from 'moment';
4  import 'react-big-calendar/lib/css/react-big-calendar.css';
5  import { initializeApp } from 'firebase/app';
6  import { getFirestore, collection, addDoc, getDocs } from 'firebase/firestore';
7  import CustomToolbar from './components/CustomToolbar';
8  import '../css/visit-form-style.css';
```

Slika 4.9. Uvoz ključnih funkcionalnosti

```
10  // Firebase configuration
11  const firebaseConfig = {
12    apiKey: "AIzaSyB9G5-XQXysma-DI5Q8NV8LYx-9EmgyxE",
13    authDomain: "beyoutiful-958a8.firebaseapp.com",
14    projectId: "beyoutiful-958a8",
15    storageBucket: "beyoutiful-958a8.appspot.com",
16    messagingSenderId: "660284348148",
17    appId: "1:660284348148:web:440c5784db1e22e12d70a1",
18    measurementId: "G-9TYRF43T87"
19  };
20
21  // Initialize Firebase
22  const app = initializeApp(firebaseConfig);
23  const db = getFirestore(app);
24  const localizer = momentLocalizer(moment);
```

Slika 4.10. Konfiguracija i inicijalizacija Firebase-a

U Scheduler komponenti (Slika 4.11.) koriste se React-ovi hookovi useState i useEffect za rukovanje podacima i njihovim promjenama unutar komponente koja upravlja prikazom i manipulacijom termina. useState se koristi za definiranje tri stanja: events, formDate, formTime i selectedEmployee. Varijabla events pohranjuje sve događaje (termine) koji će biti prikazani u kalendaru. formDate i formTime koriste se za pohranu odabranog datuma i vremena, dok selectedEmployee prati koji je zaposlenik trenutno odabran, omogućujući filtriranje događaja po zaposleniku. Unutar useEffect hooka, funkcija fetchEvents se koristi za asinkrono dohvaćanje podataka o terminima iz Firestore baze podataka. Kada se podaci dohvate, svaki termin se transformira u format prikladan za prikaz unutar kalendara. Svaki događaj uključuje start i end vremenske oznake (izrađene kombinacijom datuma i vremena početka i kraja termina), naslov koji opisuje uslugu (title), te zaposlenika koji je zadužen za taj termin (employee). Jedan ključan dio funkcionalnosti je filtriranje termina na temelju odabranog zaposlenika. Ako je zaposlenik odabran (odnosno, vrijednost selectedEmployee nije prazna), dohvaćeni događaji se filtriraju tako da prikazuju samo one termine

koji su povezani s tim zaposlenikom. Ako zaposlenik nije odabran, prikazuju se svi dostupni događaji bez filtriranja. Nakon što se događaji filtriraju prema zaposleniku, pohranjuju se u stanje events putem funkcije setEvents, koja automatski ažurira prikaz kalendara na stranici. Zahvaljujući useEffect, ovaj proces se automatski pokreće svaki put kad se promijeni vrijednost selectedEmployee, što omogućuje dinamično filtriranje događaja kada korisnik promijeni zaposlenika. Na taj način, kalendar uvijek prikazuje točne termine povezane s odabranim zaposlenikom. Funkcija handleSelectSlot (Slika 4.12.) omogućava korisniku da odabere termin u kalendaru i automatski ažurira podatke o datumu i vremenu u formi za rezervaciju.

```
141  const Scheduler = () => {
142    const [events, setEvents] = useState([]);
143    const [formDate, setFormDate] = useState('');
144    const [formTime, setFormTime] = useState('');
145    const [selectedEmployee, setSelectedEmployee] = useState('');
146
147    useEffect(() => {
         Codeium: Refactor | Explain | Generate JSDoc | ×
148      const fetchEvents = async () => {
149        const snapshot = await getDocs(collection(db, 'appointments'));
150        const fetchedEvents = snapshot.docs.map(doc => {
151          const data = doc.data();
152          return {
153            start: new Date(data.date + ' ' + data.time),
154            end: new Date(data.date + ' ' + data.endTime),
155            title: data.service,
156            employee: data.employee
157          };
158        });
159
160        const filteredEvents = selectedEmployee
161          ? fetchedEvents.filter(event => event.employee === selectedEmployee)
162          : fetchedEvents;
163
164        setEvents(filteredEvents);
165      };
166
167      fetchEvents();
168    }, [selectedEmployee]);
```

Slika 4.11. Kod Scheduler komponente

```
170   const handleSelectSlot = (slotInfo) => {
171     const startDate = moment(slotInfo.start).format('YYYY-MM-DD');
172     const startTime = moment(slotInfo.start).format('HH:mm');
173     setFormDate(startDate);
174     setFormTime(startTime);
175   };
```

Slika 4.12. Kod handleSelectSlot

4.2.2. Forma za rezervaciju

Komponenta BookingForm (Slika 4.13.) služi kao dinamički obrazac u kojem korisnik može unijeti svoje podatke kako bi rezervirao termin za odabranu uslugu. Ova forma se povezuje s odabranim datumom, vremenom i zaposlenikom putem proslijeđenih parametara (selectedDate, selectedTime, selectedEmployee), čime se omogućava dinamičko ažuriranje sadržaja obrasca prema korisnikovim izborima iz kalendara.

formData koristi useState za pohranu unosa korisnika, uključujući:

- Ime korisnika (name)
- E-mail adresu korisnika (email)
- Odabranu uslugu (service)
- Kontakt informacije (contact)
- Datum termina (date)
- Vrijeme termina (time)
- Zaposlenika koji će obaviti uslugu (employee)

Ovi podaci čine osnovu svakog zakazivanja termina i korisnik ih mora ispravno ispuniti prije nego što se podaci pošalju.

```
33  const BookingForm = ({ selectedDate, selectedTime, selectedEmployee }) => {
34    const [formData, setFormData] = useState({
35      name: '',
36      email: '',
37      service: '',
38      contact: '',
39      date: selectedDate || '',
40      time: selectedTime || '',
41      employee: selectedEmployee || ''
42    });
```

Slika 4.13. Kod BookingForm komponente

useEffect (Slika 4.14.) služi za automatsko ažuriranje stanja forme kada se promijeni bilo koji od proslijeđenih parametara (selectedDate, selectedTime, selectedEmployee). Na ovaj način, ako korisnik u kalendaru odabere novi datum, vrijeme ili zaposlenika, vrijednosti unutar obrasca se ažuriraju bez potrebe za dodatnim unosom.

```
53    useEffect(() => {
54      setFormData((prevData) => ({
55        ...prevData,
56        date: selectedDate,
57        time: selectedTime,
58        employee: selectedEmployee
59      }));
60    }, [selectedDate, selectedTime, selectedEmployee]);
```

Slika 4.14. Kod useEffect funkcije

4.2.3. Podnošenje forme i spremanje podataka

Funkcija handleSubmit (Slika 4.15.) je aktivirajuća funkcija prilikom slanja forme. Ona prvo sprječava standardno ponašanje forme (event.preventDefault()) kako bi omogućila prilagođenu obradu. Unutar funkcije, svi potrebni podaci se uzimaju iz formData. Ako bilo koje od obaveznih polja (ime, e-mail, datum, vrijeme, usluga, zaposlenik) nije popunjeno, funkcija prikazuje poruku korisniku kako bi ih obavijestila da popune sva polja. Ako su svi podaci uneseni, stvara se objekt appointmentData koji sadrži sve podatke o zakazanom terminu, uključujući izračunato vrijeme završetka termina (endTime). Ovaj izračun koristi biblioteku moment.js kako bi odabrano vrijeme početka dodalo trajanje usluge (na temelju odabranog tipa usluge). Pomoću addDoc, podaci se spremaju u Firebase bazu podataka u kolekciju appointments. Nakon uspješnog spremanja, korisniku

se prikazuje poruka o uspješnom zakazivanju termina, a obrazac se resetira kako bi bio spreman za eventualni novi unos. Kako bi se, svaki put kada korisnik unese podatke u neko od polja, ažuriralo stanje formData, koristi se funkcija handleChange (Slika 4.16.) koristeći ID polja (npr. name, email, service) i novu vrijednost unosa.

```
63  const handleSubmit = async (event) => {
64    event.preventDefault();
65    const { name, email, service, contact, date, time, employee } = formData;
66
67    if (!name || !email || !date || !time || !service || !employee) {
68      alert('Please fill in all required fields');
69      return;
70    }
71
72    const appointmentData = {
73      name,
74      email,
75      service,
76      contact,
77      date,
78      time,
79      employee,
80      endTime: moment(time, 'HH:mm').add(serviceDurations[service] || 60, 'minutes').format('HH:mm'),
81    };
82
83    try {
84      await addDoc(collection(db, 'appointments'), appointmentData);
85      alert('Appointment successfully booked!');
86
87      setFormData({
88        name: '',
89        email: '',
90        service: '',
91        contact: '',
92        date: selectedDate || '',
93        time: selectedTime || '',
94        employee: selectedEmployee || ''
95      });
96    } catch (error) {
97      console.error('Error booking appointment: ', error);
98    }
99  };
```

Slika 4.15. Kod handleSubmit funkcije

```
101 const handleChange = (e) => {
102   setFormData({ ...formData, [e.target.id]: e.target.value });
103 };
```

Slika 4.16. Kod handleChange funkcije

4.2.4. Prikaz forme

Obrazac za prikaz forme (Slika 4.17.) se sastoji od više polja za unos. Prvi dio obrasca uključuje dva tekstualna polja: jedno za unos imena korisnika i jedno za unos e-mail adrese. Ta polja koriste onChange događaj za ažuriranje podataka u formData objektu pomoću handleChange funkcije. Oba polja su označena kao obavezna (required). Sljedeći dio obrasca sadrži padajući izbornik (<select>) s opcijama usluga koje korisnik može odabrati, poput "Nails installation", "Makeup" i drugih. Ovaj izbornik također koristi onChange događaj kako bi se pratile promjene u odabiru usluge. Prva opcija je neaktivna (disabled), služi kao poruka za korisnika da odabere uslugu. Polja za datum, vrijeme i zaposlenika su samo za čitanje (readOnly), jer se ti podaci automatski ažuriraju na temelju odabira korisnika u kalendaru. Na kraju se nalazi gumb za slanje forme. Kada korisnik klikne na "Submit", aktivira se funkcija handleSubmit koja obrađuje podatke i šalje ih u Firebase. Ova forma je vrlo interaktivna i povezana s dinamičkim komponentama poput kalendara, osiguravajući jednostavno i intuitivno iskustvo zakazivanja termina za korisnike.

```
return (
  <form className="booking-form" onSubmit={handleSubmit}>
    <h3>To book a visit please fill out this form!</h3>
    <div className="inputBox">
      <input type="text" id="name" placeholder="Your name" value={formData.name} onChange={handleChange} required />
    </div>
    <div className="inputBox">
      <input type="email" id="email" placeholder="Your e-mail" value={formData.email} onChange={handleChange} required />
    </div>
    <div className="inputBox">
      <select id="service" className="custom-select" value={formData.service} onChange={handleChange} required>
        <option value="" disabled>Select a service</option>
        <option value="Nails installation">Nails installation</option>
        <option value="Nails correction">Nails correction</option>
        <option value="Eyebrow treatment">Eyebrow treatment</option>
        <option value="Depilation">Depilation</option>
        <option value="Makeup">Makeup</option>
        <option value="Solarium">Solarium</option>
      </select>
    </div>
    <div className="inputBox">
      <input type="text" id="date" value={formData.date} placeholder="Date" readOnly />
    </div>
    <div className="inputBox">
      <input type="text" id="time" value={formData.time} placeholder="Time" readOnly />
    </div>
    <div className="inputBox">
      <input type="text" id="employee" value={formData.employee} placeholder="Employee" readOnly />
    </div>
    <button type="submit" className="submit-form">Submit</button>
  </form>
);
```

Slika 4.17. Kod obrasca za prikaz forme za rezervaciju termina

4.3. Pisanje recenzije

Korisnicima koji žele ostaviti recenziju to je omogućeno u posljednjoj sekciji HTML-a. Sekcija (Slika 4.18.) se sastoji od prikaza recenzija, gumba za dodavanje nove recenzije, i modala (skočnog prozora) u kojem korisnik može unijeti svoju recenziju. Sekcija s klasom review prikazuje naslov „Customer's Review“ i gumb za dodavanje nove recenzije. Gumb za dodavanje recenzije (add-review-btn) kada se klikne, otvara modal za unos nove recenzije. div s ID-om reviews-container je prazan, a njegova svrha je da se recenzije generirane putem JavaScript-a dinamički umetnu unutar njega. Kada korisnici dodaju nove recenzije, one će se ovdje prikazati.

```
201  <section class="review" id="review">
202      <h1 class="heading">Customer's Review</h1>
203      <button id="add-review-btn">Add Review</button>
204
205      <div class="box-container" id="reviews-container">
206      </div>
207  </section>
```

Slika 4.18. Kod sekcije za ostavljanje recenzije

Modal (Slika 4.19.) je skriven dok se ne otvori klikom na gumb „Add Review“. Ima klasu modal i modal-content za stiliziranje sadržaja modala. Gumb za zatvaranje modala (close-review-modal) je označen simbolom „×“ koji omogućava zatvaranje modala bez unosa recenzije. Unutar modala nalazi se obrazac (form) s ID-om review-form koji omogućava korisniku unos recenzije. Korisnik može unijeti svoje ime u tekstualno polje (reviewer-name), tekst recenzije u tekstualno područje (review-textbox), te ocjenu usluge putem padajućeg izbornika (review-rating) s ocjenama od 1 do 5. Gumb „Submit Review“ (submit-review-btn) šalje unesene podatke kada korisnik završi s unosom.

```
<div id="review-modal" class="modal">
    <div class="modal-content">
        <span class="close" id="close-review-modal">&times;</span>
        <h1>Add Your Review</h1>
        <form id="review-form">
            <input type="text" id="reviewer-name" placeholder="Your Name" required>
            <textarea id="review-textbox" placeholder="Your Review" required></textarea>
            <select id="review-rating" required>
                <option value="" disabled selected>Rating</option>
                <option value="5">5 - Excellent</option>
                <option value="4">4 - Very Good</option>
                <option value="3">3 - Good</option>
                <option value="2">2 - Fair</option>
                <option value="1">1 - Poor</option>
            </select>
            <button type="submit" id="submit-review-btn">Submit Review</button>
        </form>
    </div>
</div>
```

Slika 4.19. Modal za unos recenzije

4.3.1. JavaScript za dodavanje recenzija

JavaScript kod omogućava dinamičko dohvaćanje, prikaz i unos recenzija na web stranici koristeći Firestore bazu podataka. Kod pokriva nekoliko ključnih funkcionalnosti: dohvaćanje najnovijih recenzija, unos novih recenzija, te upravljanje modalnim prozorom za unos recenzija. Funkcija fetchLatestReviews (Slika 4.20.) dohvaća zadnjih pet recenzija iz Firestore baze podataka, poredanih prema vremenu unosa (timestamp) u silaznom redoslijedu (najnovije na vrhu). Svaka recenzija se dinamički kreira unutar div elementa s klasom box. Unutar tog elementa prikazuju se ime recenzenta, tekst recenzije i ocjena u obliku zvjezdica. Zvjezdice se generiraju na temelju ocjene (rating), gdje se za svaku jedinicu ocjene dodaje jedna zvjezdica. Nakon što se kreira recenzija, ona se dodaje u HTML element s ID-om reviews-container, čime se recenzije prikazuju na stranici. Funkcija je povezana s dohvaćanjem podataka putem Firebase Firestore API-ja. Dio koda na slici 4.21. omogućava korisnicima dodavanje novih recenzija putem forme. Kada korisnik pritisne gumb „Submit“, funkcija sprječava zadano ponašanje forme pomoću e.preventDefault(). Iz forme se uzimaju ime korisnika, tekst recenzije i ocjena, te se stvara vremenski pečat (engl. *timestamp*) s trenutnim vremenom. Podaci se zatim spremaju u Firestore bazu podataka u kolekciju reviews. Nakon što se recenzija uspješno pošalje, korisniku se prikazuje poruka o uspješnom dodavanju, popis recenzija se ponovno dohvaća (osvježava), a forma se resetira kako bi bila spremna za sljedeći unos.

```
function fetchLatestReviews() {
    const reviewsContainer = document.getElementById('reviews-container');
    reviewsContainer.innerHTML = '';

    db.collection('reviews')
      .orderBy('timestamp', 'desc')
      .limit(5)
      .get()
      .then(querySnapshot => {
        querySnapshot.forEach(doc => {
          const reviewData = doc.data();

          const reviewBox = document.createElement('div');
          reviewBox.classList.add('box');

          const starsDiv = document.createElement('div');
          starsDiv.classList.add('stars');
          for (let i = 0; i < reviewData.rating; i++) {
            const star = document.createElement('i');
            star.classList.add('fas', 'fa-star');
            starsDiv.appendChild(star);
          }

          reviewBox.innerHTML = `
            <h3>${reviewData.name}</h3>
            <p>${reviewData.review}</p>
          `;
          reviewBox.appendChild(starsDiv);

          // Dodaj recenziju u container
          reviewsContainer.appendChild(reviewBox);
        });
      })
      .catch(error => {
        console.error('Error fetching reviews:', error);
      });
}
```

Slika 4.20. Kod fetchLatestReviews funkcije

```
61  document.getElementById('review-form').addEventListener('submit', function (e) {
62    e.preventDefault();
63
64    const name = document.getElementById('reviewer-name').value;
65    const review = document.getElementById('review-textbox').value;
66    const rating = parseInt(document.getElementById('review-rating').value);
67    const timestamp = new Date();
68
69    db.collection('reviews').add({
70      name: name,
71      review: review,
72      rating: rating,
73      timestamp: timestamp
74    }).then(() => {
75      alert('Recenzija uspješno dodana!');
76      fetchLatestReviews();
77      document.getElementById('review-form').reset();
78    }).catch(error => {
79      console.error('Greška prilikom dodavanja recenzije:', error);
80    });
81  });
```

Slika 4.21. Kod forme za dodavanje novih recenzija

4.3.2. JavaScript za upravljanje modalom unosa recenzije

Modal je skočni prozor u kojem korisnik može unijeti svoju recenziju. Modal (Slika 4.22.) se otvara kada korisnik klikne na gumb „Add Review“, a zatvara se klikom na gumb „×“ ili klikom izvan područja modala. U prvom dijelu, tri elementa iz DOM-a dohvaćaju se pomoću metoda getElementById i pohranjuju u varijable: addReviewBtn za gumb koji otvara modalni prozor, reviewModal za sam modalni prozor, i closeReviewModalBtn za gumb koji zatvara modal. Zatim se dodaju *click event listeneri* na gumbe i prozor. Prvi *event listener* se povezuje s addReviewBtn i služi za otvaranje modala. Kada se na ovaj gumb klikne, stil modala se mijenja tako da se postavi na "block", što ga čini vidljivim. Drugi *event listener* je za gumb closeReviewModalBtn, koji zatvara modal mijenjajući njegov stil na "none", čineći ga nevidljivim. Treći *event listener* se dodaje na cijeli prozor (engl. *window*). On osluškuje klikove izvan modala kako bi ga zatvorio. Provjerava se je li ciljani element događaja (event.target) jednak modalnom prozoru (reviewModal). Ako je, zatvara modal postavljanjem stila na "none". Ova funkcionalnost omogućuje zatvaranje modala kada se klikne izvan njegovog područja.

```
const addReviewBtn = document.getElementById('add-review-btn');
const reviewModal = document.getElementById('review-modal');
const closeReviewModalBtn = document.getElementById('close-review-modal');

addReviewBtn.addEventListener('click', function() {
    reviewModal.style.display = 'block';
});

closeReviewModalBtn.addEventListener('click', function() {
    reviewModal.style.display = 'none';
});

window.addEventListener('click', function(event) {
    if (event.target === reviewModal) {
        reviewModal.style.display = 'none';
    }
});
```

Slika 4.22. Kod upravljanja modalom za unos recenzije

4.4. Admin dashboard

Kako bi vlasnik i radnici na jednome mjestu mogli imati uvid u sve termine, upravljati njima i pogledati sve recenzije, uvedena je mogućnost prijave u sučelje namijenjeno samo njima. U navigaciji pritiskom na gumb admin, otvara se modal (Slika 4.23.) koji omogućuje prijavu u admin sučelje.

```
<div id="admin-modal" class="modal">
    <div class="modal-content">
        <span class="close" id="close-admin-modal">&times;</span>
        <h1>Admin Login</h1>
        <form id="admin-login-form">
            <input type="email" id="admin-email" placeholder="Email" required>
            <input type="password" id="admin-password" placeholder="Password" required>
            <button type="submit">Login</button>
        </form>
    </div>
</div>
```

Slika 4.23. Kod modala za prijavu u admin dashboard

4.4.1. Implementacija admin dashboarda

Komponenta AdminDashboard (Slika 4.24.) koristi React hooks kako bi upravljala stanjem unutar komponente. Različita stanja se definiraju pomoću useState hooka, koji omogućava pohranjivanje vrijednosti u stanju i ažuriranje tih vrijednosti kroz komponentu.

Stanja i funkcije komponente AdminDashboard:

- events je stanje koje pohranjuje sve događaje (termine) koje dohvaćamo iz Firestorea (početna lista svih termina koji su pohranjeni u kolekciji appointments na Firestoreu)
- useState**([])** postavlja inicijalnu vrijednost stanja na prazan niz [], jer kad se komponenta prvi put učita, još nemamo dohvaćene događaje
- setEvents je funkcija koja ažurira stanje events (kada dohvatimo termine iz baze, koristimo ovu funkciju kako bismo pohranili sve događaje unutar ove varijable)
- stanje koje pohranjuje trenutno odabrani događaj (termin) naziva se selectedEvent (kada korisnik klikne na neki događaj u kalendaru, taj događaj se pohranjuje ovdje)
- useState(null) inicijalizira stanje s vrijednošću null, što znači da na početku nije odabran niti jedan događaj
- setSelectedEvent je funkcija koja se koristi za postavljanje vrijednosti odabranog događaja (kada korisnik klikne na događaj, poziva se ova funkcija kako bi pohranila podatke o tom događaju)
- detailsModalIsOpen kontrolira vidljivost modala za prikaz detalja o odabranom terminu (boolean vrijednost koja određuje je li modal s detaljima otvoren ili zatvoren)
- useState(false) inicijalno postavlja modal na zatvoreno stanje
- setDetailsModalIsOpen je funkcija koja se koristi za otvaranje ili zatvaranje ovog modala
- setNewAppointment je funkcija koja ažurira stanje newAppointment. Ova funkcija se koristi za pohranjivanje vrijednosti koje korisnik unosi u formu
- selectedEmployee pohranjuje trenutno odabranog zaposlenika iz padajućeg izbornika. Kada korisnik odabere zaposlenika, njegovo ime se pohranjuje u ovo stanje

```
42  const AdminDashboard = () => {
43    const [events, setEvents] = useState([]);
44    const [selectedEvent, setSelectedEvent] = useState(null);
45    const [modalIsOpen, setModalIsOpen] = useState(false);
46    const [detailsModalIsOpen, setDetailsModalIsOpen] = useState(false);
47    const [newAppointment, setNewAppointment] = useState({
48      name: '',
49      email: '',
50      contact: '',
51      service: '',
52      date: '',
53      time: '',
54      employee: ''
55    });
56    const [selectedEmployee, setSelectedEmployee] = useState('');
```

Slika 4.24. Kod AdminDashboard komponente

Unutar useEffect hooka komponente AdminDashboard (Slika 4.25.) definira se funkcija fetchEvents koja dohvaća sve termine iz kolekcije appointments u Firestore bazi podataka. getDocs dohvaća sve dokumente u kolekciji, zatim se svaki dokument mapira u JavaScript objekt koji sadrži podatke termina, poput početnog i završnog vremena, tipa usluge (naslova) i zaposlenika. Nakon što su podaci dohvaćeni, pohranjuju se u stanje events pomoću setEvents.

```
58  useEffect(() => {
        Codeium: Refactor | Explain | Generate JSDoc
59      const fetchEvents = async () => {
60      const snapshot = await getDocs(collection(db, 'appointments'));
61      const fetchedEvents = snapshot.docs.map(doc => {
62        const data = doc.data();
63        return {
64          id: doc.id,
65          start: new Date(data.date + ' ' + data.time),
66          end: new Date(data.date + ' ' + data.endTime),
67          title: data.service,
68          employee: data.employee,
69          ...data,
70        };
71      });
72      setEvents(fetchedEvents);
73    };
74    fetchEvents();
75  }, []);
```

Slika 4.25. Kod useEffect hooka komponente AdminDashboard

handleDelete (Slika 4.26.) je asinkrona funkcija koja se koristi za brisanje odabranog termina iz Firestore baze podataka. Funkcija prvo provjerava je li neki događaj odabran (selectedEvent). Ako je odabran, koristi try...catch blok za izvršavanje brisanja i hvatanje eventualnih grešaka. Unutar try bloka, funkcija poziva deleteDoc, koja asinkrono briše termin iz kolekcije 'appointments' u bazi podataka, koristeći ID odabranog događaja (selectedEvent.id). Ako je brisanje uspješno, pojavljuje se obavijest ('Appointment deleted successfully!'). Zatim se zatvara modal s detaljima termina pozivanjem setDetailsModalIsOpen(false). Sljedeći korak je dohvat svih preostalih termina iz baze podataka. Poziva se getDocs funkcija na kolekciji 'appointments' i pohranjuje vraćene dokumente u snapshot varijablu. Zatim se pomoću map funkcije prolazi kroz svaki dokument iz snapshot-a kako bi se kreirao novi niz objekata fetchedEvents. Unutar svakog objekta, informacije iz baze se formatiraju: postavlja se id, početak (start) i kraj (end) termina pomoću Date objekata, te se dodaju ostali podaci poput service i employee. Nakon kreiranja niza fetchedEvents, poziva se setEvents kako bi se ažuriralo stanje s novim nizom događaja. U slučaju greške prilikom brisanja termina, catch blok hvata grešku i ispisuje je u konzolu (console.error).

Dodavanje novog termina omogućeno je asinronom handleAddAppointment funkcijom (Slika 4.27.). Funkcija počinje blokiranjem zadane radnje događaja pomoću e.preventDefault(), kako bi se spriječilo ponovno učitavanje stranice prilikom podnošenja obrasca. Zatim se iz objekta newAppointment izvlače podaci o imenu, e-mailu, kontaktu, usluzi, datumu, vremenu i zaposleniku. Provjerava se jesu li svi potrebni podaci prisutni. Ako neki od njih nedostaje, prikazuje se upozorenje korisniku i funkcija se zaustavlja. Sljedeći korak je određivanje trajanja usluge pomoću serviceDurations objekta, gdje se prema odabranoj usluzi određuje trajanje u minutama. Ako trajanje nije unaprijed definirano, koristi se zadana vrijednost od 60 minuta. Pomoću biblioteke moment, izračunava se endTime na osnovu početnog vremena i trajanja usluge, te se formatira u oblik 'HH:mm'. Nakon toga, u try bloku, funkcija poziva addDoc kako bi dodala novi dokument u kolekciju 'appointments' u bazi podataka, prosljeđujući unesene podatke, uključujući i izračunato vrijeme završetka (endTime). Ako je dodavanje uspješno, prikazuje se obavijest korisniku. Modalni prozor za unos termina se zatvara pozivom setModalIsOpen(false), a obrazac se resetuje na početne vrijednosti pozivanjem setNewAppointment. Nakon dodavanja termina, dohvaćaju se svi postojeći termini iz baze podataka pomoću getDocs. Zatim se kroz mapfunkciju prolazi kroz svaki dokument kako bi se kreirao niz objekata fetchedEvents, gdje se formatiraju podaci poput ID-a, početnog i krajnjeg vremena, naslova i zaposlenika. Nakon kreiranja niza događaja, poziva se setEvents kako bi se

ažurirao prikaz termina. U slučaju greške prilikom dodavanja termina, catch blok hvata grešku i ispisuje je u konzolu.

```
86  const handleDelete = async () => {
87    if (selectedEvent) {
88      try {
89        await deleteDoc(doc(db, 'appointments', selectedEvent.id));
90        alert('Appointment deleted successfully!');
91        setDetailsModalIsOpen(false);
92        const snapshot = await getDocs(collection(db, 'appointments'));
93        const fetchedEvents = snapshot.docs.map(doc => {
94          const data = doc.data();
95          return {
96            id: doc.id,
97            start: new Date(data.date + ' ' + data.time),
98            end: new Date(data.date + ' ' + data.endTime),
99            title: data.service,
100           employee: data.employee,
101           ...data,
102         };
103       });
104       setEvents(fetchedEvents);
105     } catch (error) {
106       console.error('Error deleting appointment: ', error);
107     }
108   }
109 };
```

Slika 4.26. Kod handleDelete funkcije

```
const handleAddAppointment = async (e) => {
  e.preventDefault();
  const { name, email, contact, service, date, time, employee } = newAppointment;
  if (!name || !email || !contact || !service || !date || !time || !employee) {
    alert('Please fill in all fields.');
    return;
  }
  const duration = serviceDurations[service] || 60;
  const endTime = moment(time, 'HH:mm').add(duration, 'minutes').format('HH:mm');
  try {
    await addDoc(collection(db, 'appointments'), {
      name,
      email,
      contact,
      service,
      date,
      time,
      endTime,
      employee,
    });
    alert('Appointment added successfully!');
    setModalIsOpen(false);
    setNewAppointment({ name: '', email: '', contact: '', service: '', date: '', time: '', employee: '' });
    const snapshot = await getDocs(collection(db, 'appointments'));
    const fetchedEvents = snapshot.docs.map(doc => {
      const data = doc.data();
      return {
        id: doc.id,
        start: new Date(data.date + ' ' + data.time),
        end: new Date(data.date + ' ' + data.endTime),
        title: data.service,
        employee: data.employee,
        ...data,
      };
    });
    setEvents(fetchedEvents);
  } catch (error) {
    console.error('Error adding appointment: ', error);
  }
};
```

Slika 4.27. Kod handleAddAppointment funkcije

4.5. Ključni CSS dijelovi

U ovoj web aplikaciji CSS je strukturiran na način koji omogućuje konzistentan izgled elemenata, responzivnost za različite uređaje, te intuitivnu interakciju s korisnikom. U nastavku ćemo prikazati najvažnije i najutjecajnije dijelove ovog CSS-a, koji oblikuju izgled i funkcionalnost aplikacije.

Na početku CSS-a definirane su globalne varijable za boje i stilove. Kako prikazuje slika 4.28., varijable postavljaju glavne boje, boje pozadine, granice i sjene koje su korištene za konzistentnost i lakšu promjenu stila na cijeloj web stranici. Stil selektora * primjenjuje globalne postavke na sve elemente na stranici. Postavlja se zadani font 'Playfair Display' za cijelu stranicu, koji je uparen sa sans-serif kao rezervnim fontom u slučaju da primarni nije dostupan. Uz to, uklanjaju se svi *margin*

i *padding* postavke za elemente kako bi se izbjegle nepredviđene praznine i osigurao kontrolirani razmak između elemenata. Postavljanje box-sizing svojstva na border-box omogućava da veličine elemenata uključuju *padding* i *margin*, čime je lakše upravljati njihovim dimenzijama. Element html je stiliziran tako da ima osnovnu veličinu fonta od 10px. Također, postavljen je overflow-x: hidden, što sprječava vodoravno pomicanje stranice ako neki elementi prelaze preko širine prikaza. scroll-behavior: smooth omogućava glatko pomicanje stranice kad korisnik koristi linkove za unutarnju navigaciju. Za sekcije, postavljen je *padding* od 5rem s gornje i donje strane, te 7% sa strane. Ovaj razmak osigurava da sadržaj unutar sekcija ne bude preblizu rubovima ekrana i pruža ugodan vizualni prostor.

```
3   :root{
4
5       --main-color:#d7c7ae;
6       --primary-color: #8a6d46;
7       --hover: #5a4d3e;
8       --black:#2a201c;
9       --white:#fff;
10      --light-bg:#f3efec;
11      --dark-bg:rgba(0,0,0,.7);
12      --border:.1rem solid #aaa;
13      --box-shadow:0 .5rem 1rem rgba(0,0,0,.1);
14  }
15
16  *{
17      font-family: 'Playfair Display', sans-serif;
18      margin:0; padding:0;
19      box-sizing: border-box;
20      outline: none;
21      border:none;
22      text-decoration: none;
23      transition: all .2s linear;
24  }
25
26
27  html{
28      font-size: 10px;
29      overflow-x: hidden;
30      scroll-behavior: smooth;
31      scroll-padding-top: 6rem;
32  }
```

Slika 4.28. Početak CSS-a

Kako bi se poboljšalo korisničko iskustvo i web aplikacija učinila responzivnom, definirani su *media queries* (slika 4.29.). Oni omogućuju prilagodbu izgleda stranice na različitim uređajima, osiguravajući da se sadržaj automatski smanjuje, a elementi poput navigacijskog izbornika postaju

funkcionalniji i intuitivniji na manjim ekranima. Media query za uređaje sa širinom ekrana manjom od ili jednakom 768px, što odgovara tabletima i pametnim telefonima, prikazan je na slici 4.29. *Query* prilagođava navigaciju u zaglavlju tako da postaje prilagođena mobilnim uređajima. Gumb #menu-btn postaje vidljiv kao inline-block, što znači da se sada prikazuje na ekranu kako bi korisnici mogli otvoriti mobilni izbornik. Navigacijski izbornik unutar .header .navbar koristi position: fixed, što znači da je postavljen fiksno na vrh stranice i zauzima cijelu visinu ekrana (height: 100%). Izbornik koristi flexbox kako bi stavke unutar njega bile vertikalno poravnate i centrirane. .header .navbar.active mijenja right vrijednost u 0 kada je izbornik aktivan, čime se omogućuje da izbornik klizi s desne strane u vidljivi dio ekrana. Unutar izbornika, linkovi su stilizirani tako da su blok elementi (zauzimaju cijelu širinu), s centriranim tekstom i povećanom veličinom fonta (3rem) kako bi bili lakše čitljivi na manjim ekranima.

Ostatak CSS-a se uglavnom bavi podešavanjem osnovnih stilskih elemenata kao što su boja, pozadina, veličina fonta i margina za različite sekcije stranice. Stilovi su konzistentno prilagođeni kako bi se stvorila ugodna vizualna hijerarhija i osigurao responzivan dizajn koji odgovara različitim veličinama ekrana, a bolji uvid u to kakav je izgled same stranice dobit ćemo u sljedećem poglavlju.

```
@media (max-width:768px){
    #menu-btn{
        display: inline-block;
    }
     .header .navbar {
        position: fixed;
        top: 0;
        right: -105%;
        width: 30rem;
        background: var(--white);
        height: 100%;
        display: -webkit-box;
        display: -ms-flexbox;
        display: flex;
        -webkit-box-orient: vertical;
        -webkit-box-direction: normal;
            -ms-flex-flow: column;
                flex-flow: column;
        -webkit-box-pack: center;
            -ms-flex-pack: center;
                justify-content: center;
        z-index: 1200;
      }
      .header .navbar #close-navbar {
        display: block;
      }
      .header .navbar.active {
        right: 0;
      }
      .header .navbar a {
        display: block;
        margin: 1rem 0;
        text-align: center;
        font-size: 3rem;
      }
    .home .content h3{
        font-size: 5.5rem;
    }
}
```

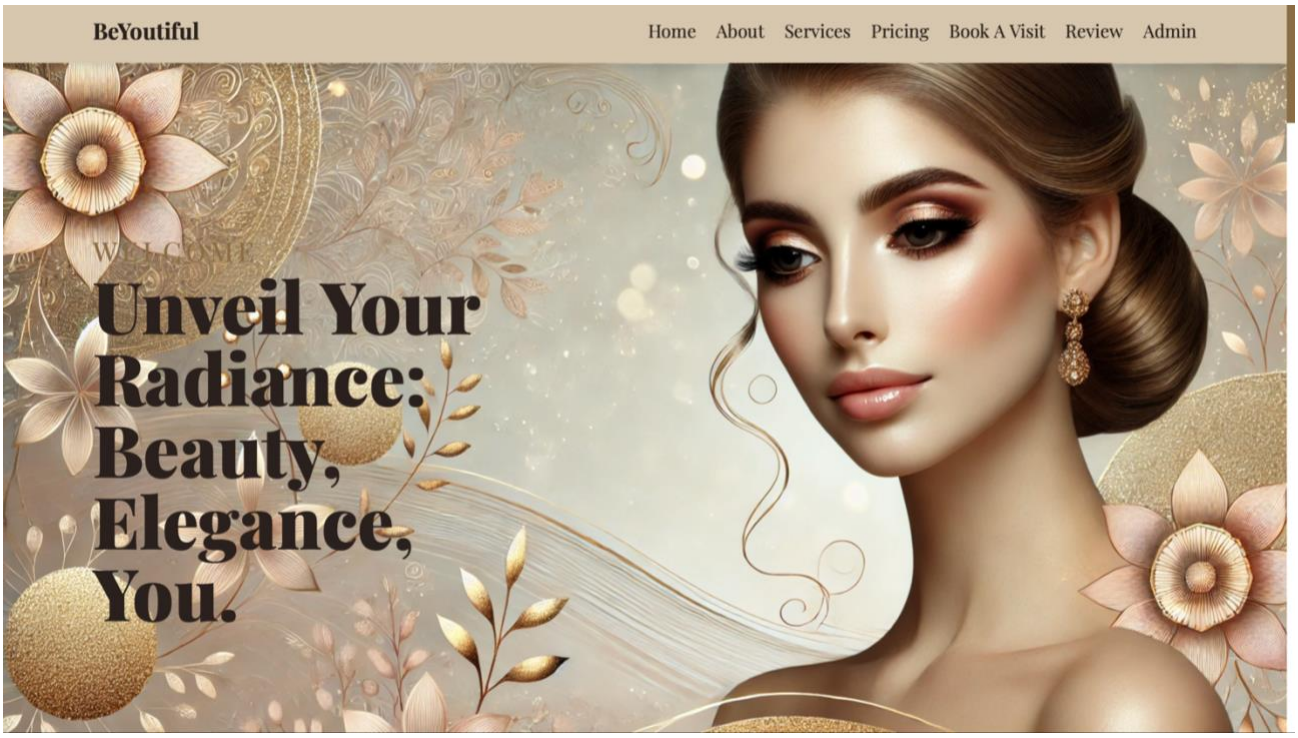
Slika 4.29. Kod media queryja za uređaje sa širinom manjom ili jednakom 768px

5. TESTIRANJE RADA APLIKACIJE

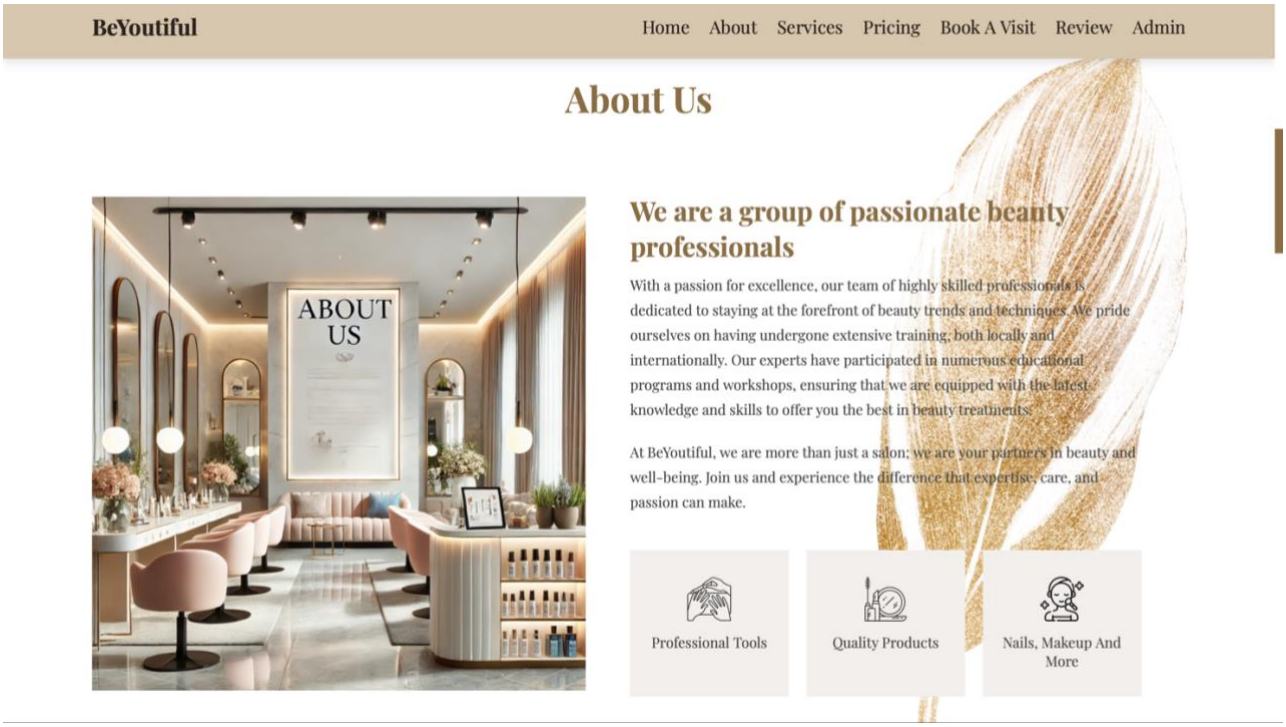
U ovom poglavlju, testirane su funkcionalnosti aplikacije uz prikaz izgleda same web aplikacije.

5.1. Izgled aplikacije

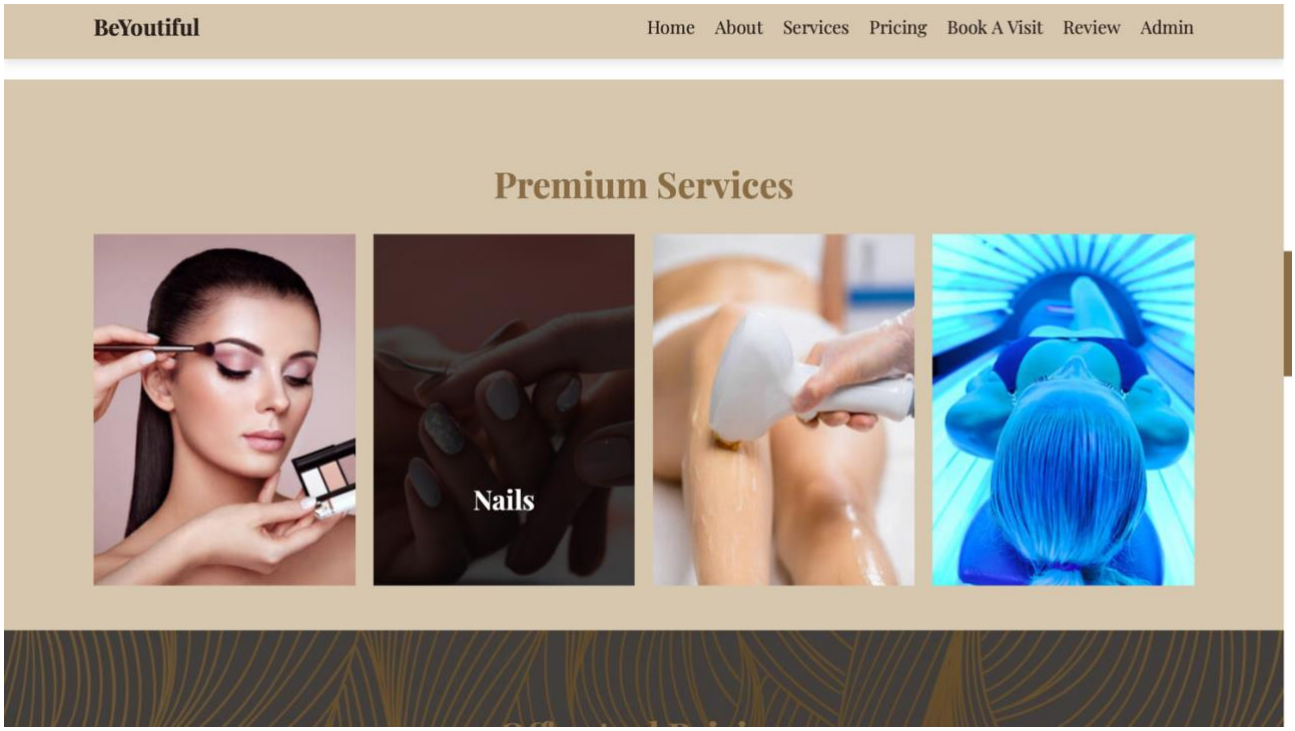
Sljedeće slike prikazuju redom izgled pojedine sekcije pritiskom na odgovarajuću tipku u navigaciji. Samo pokretanje aplikacije tj. „Home“ prikazano je na slici 5.1. Slijedi sekcija „O nama“ (Slika 5.2.), sekcija „Usluge“ (Slika 5.3.), i „Cjenik“ (Slika 5.4.).



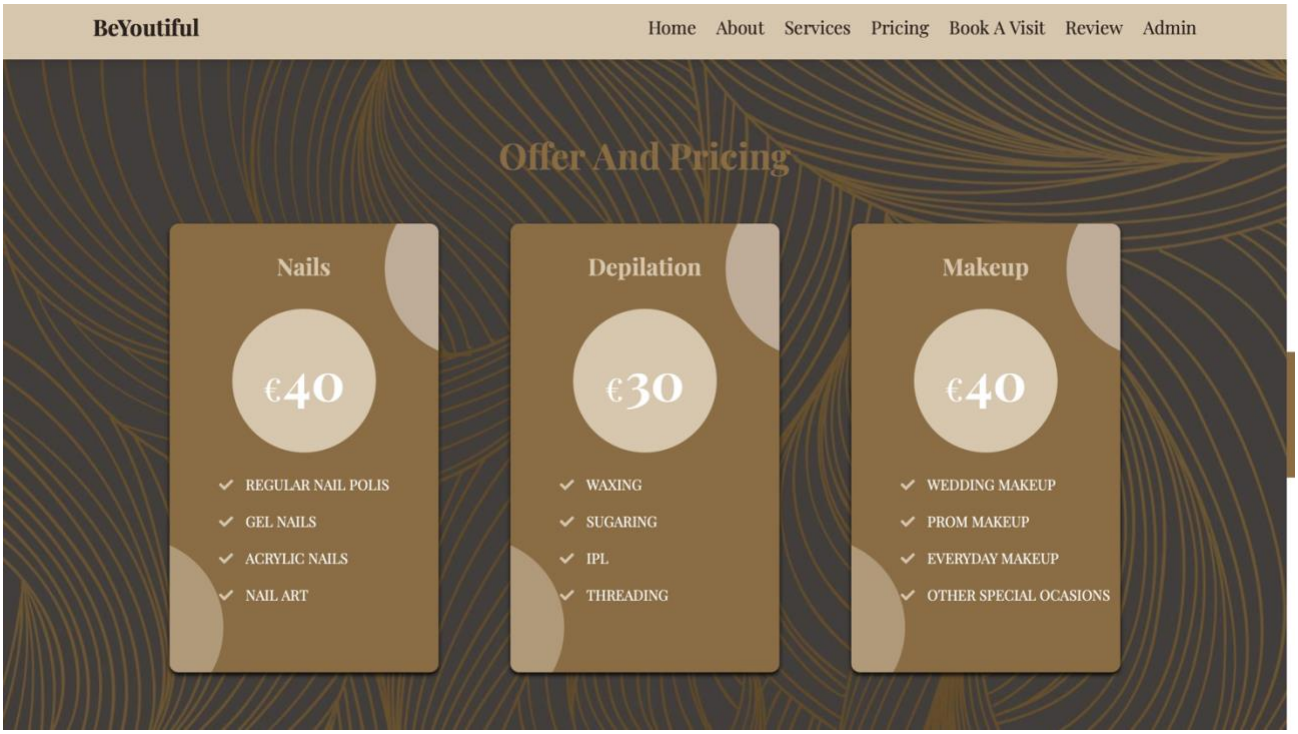
Slika 5.1. Početni prikaz web aplikacije



Slika 5.2. Sekcija „O nama“



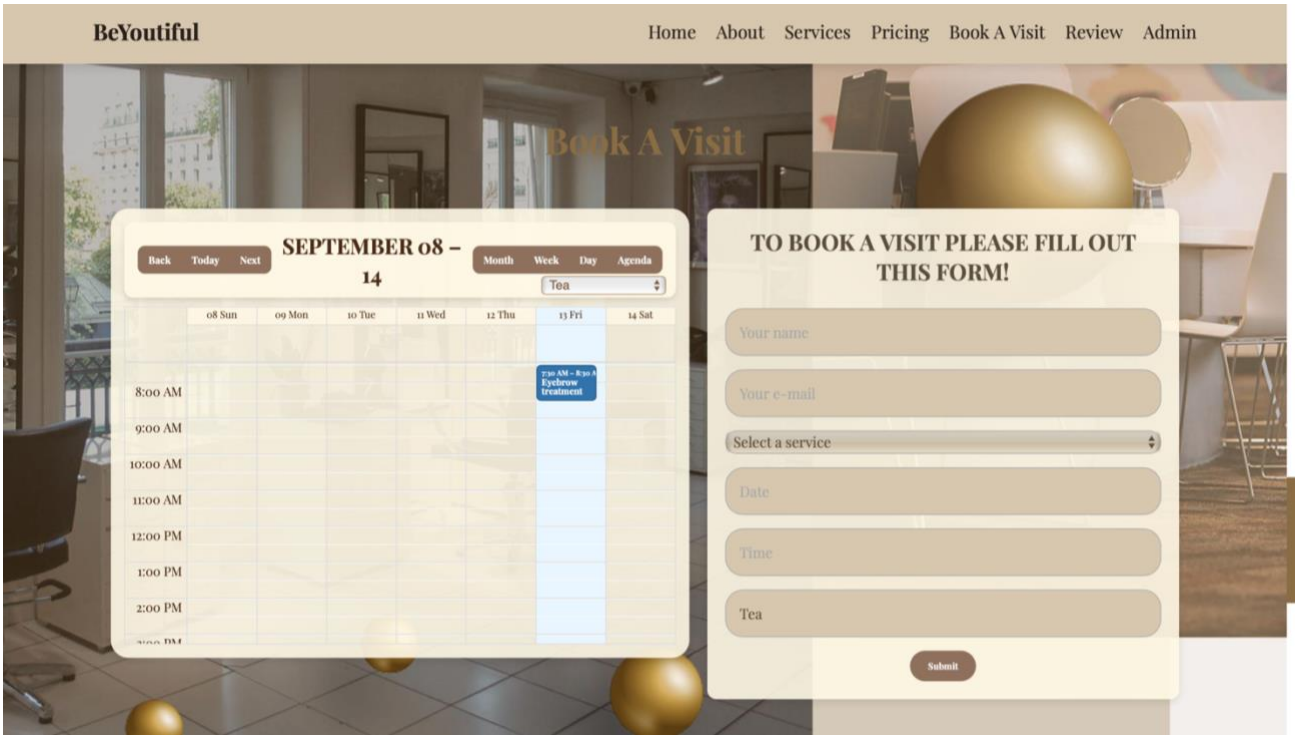
Slika 5.3. Sekcija „Usluge“



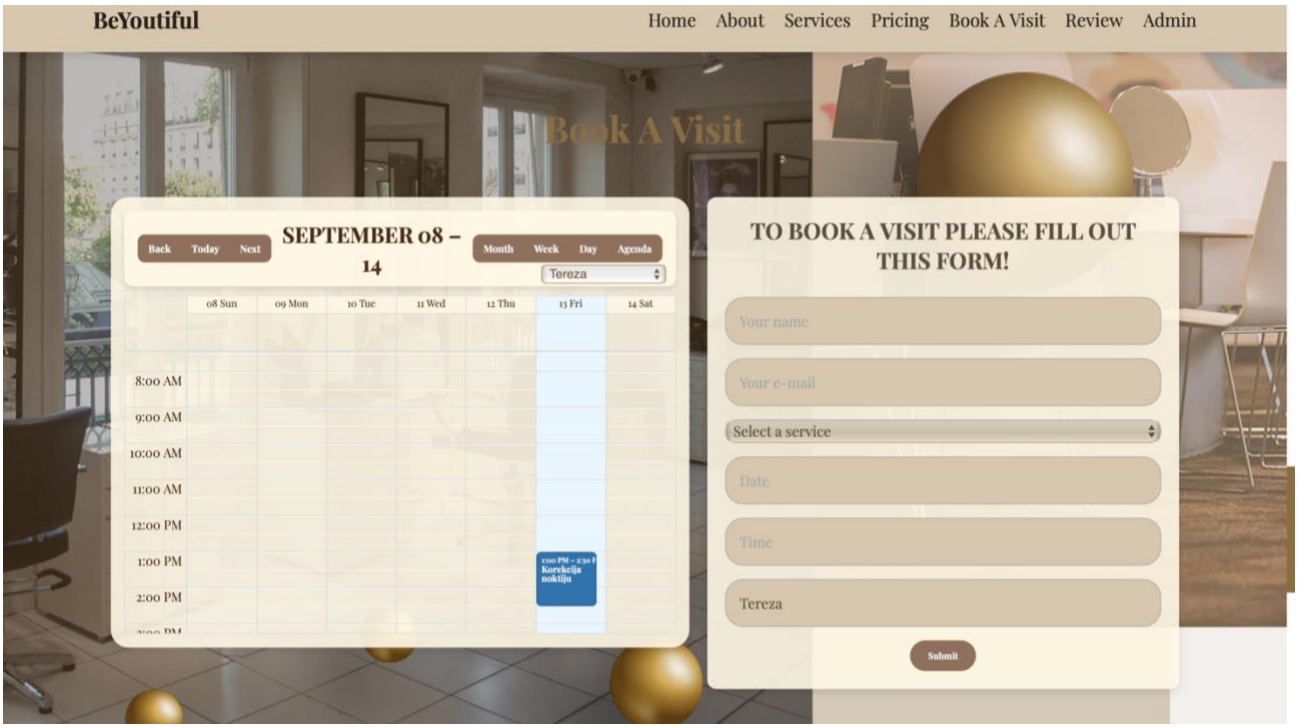
Slika 5.4. Sekcija „Cjenik“

5.2. Testiranje rezervacije termina

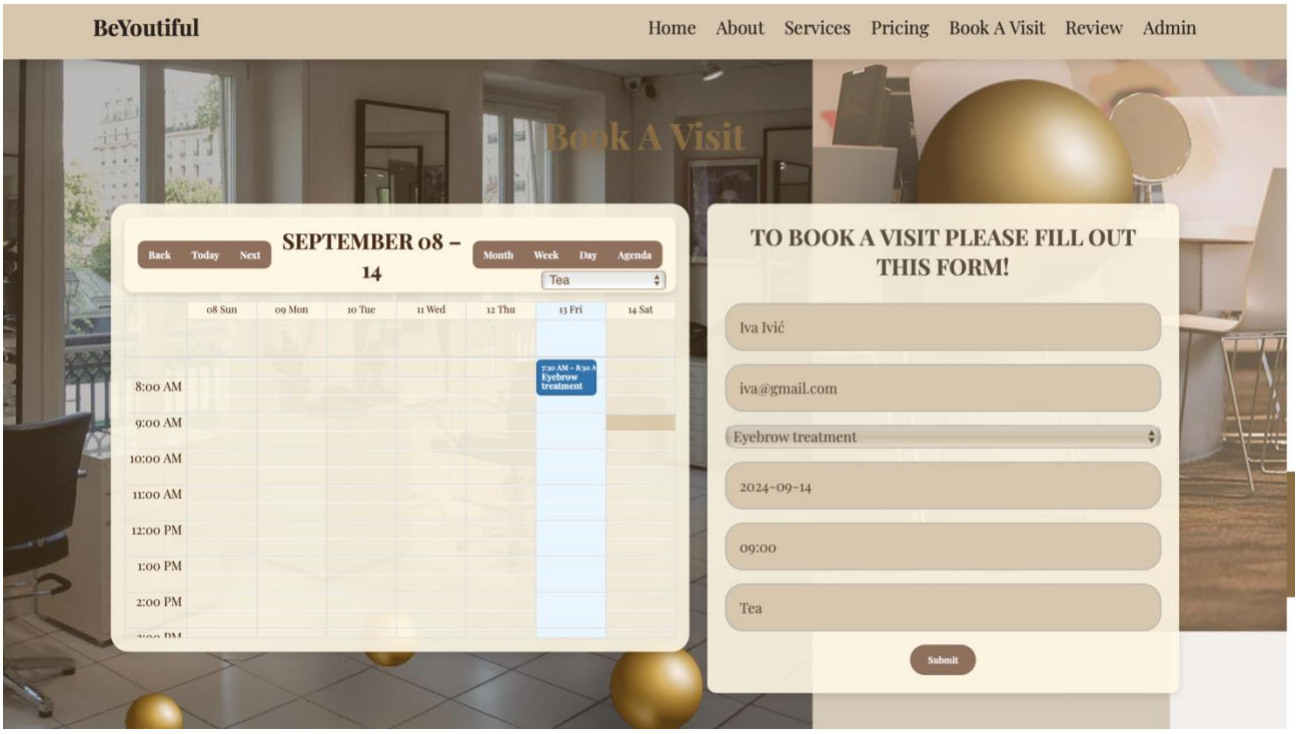
Slike 5.5. i 5.6. prikazuju prvi prikaz kada korisnik pritisne na gumb koji ga vodi ka rezervaciji termina. Na slikama možemo vidjeti različite prikaze kalendara u ovisnosti od toga koji zaposlenik je izabran, drugim riječima, prikazuju se samo termini izabranog zaposlenika. Desno od kalendara nalazi se forma koju je potrebno ispuniti ako korisnik želi rezervirati termin. Na kalendaru je moguće odabrati datum i vrijeme koji je slobodan i odgovara korisniku. Taj datum i vrijeme se automatski ispunjava u formi kako prikazuje slika 5.7. Nakon podnošenja forme pritiskom na gumb „submit“ pojavljuje se poruka koja govori o uspješnosti rezervacije (Slika 5.8.). Forma se zatim resetira, a u kalendaru možemo vidjeti kako se novi termin uspješno prikazuje u kalendaru (Slika 5.9.).



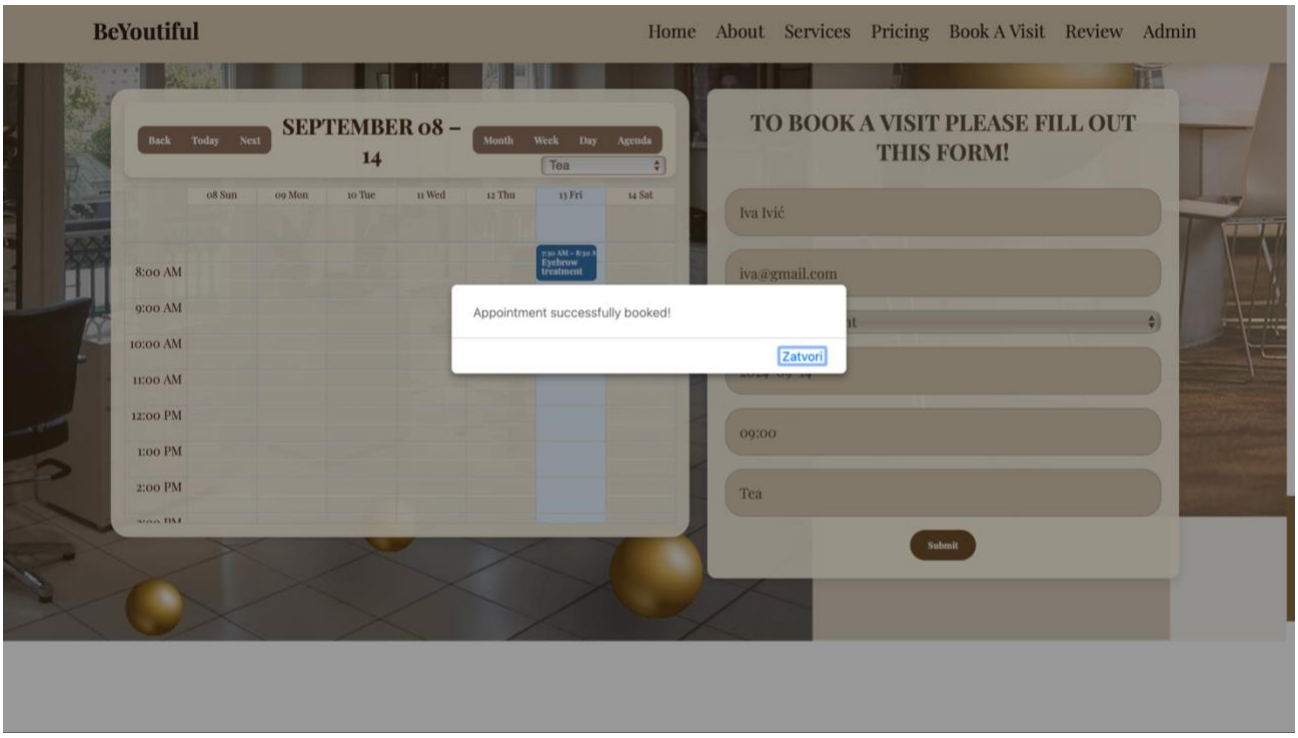
Slika 5.5. Prikaz kalendara prvog zaposlenika



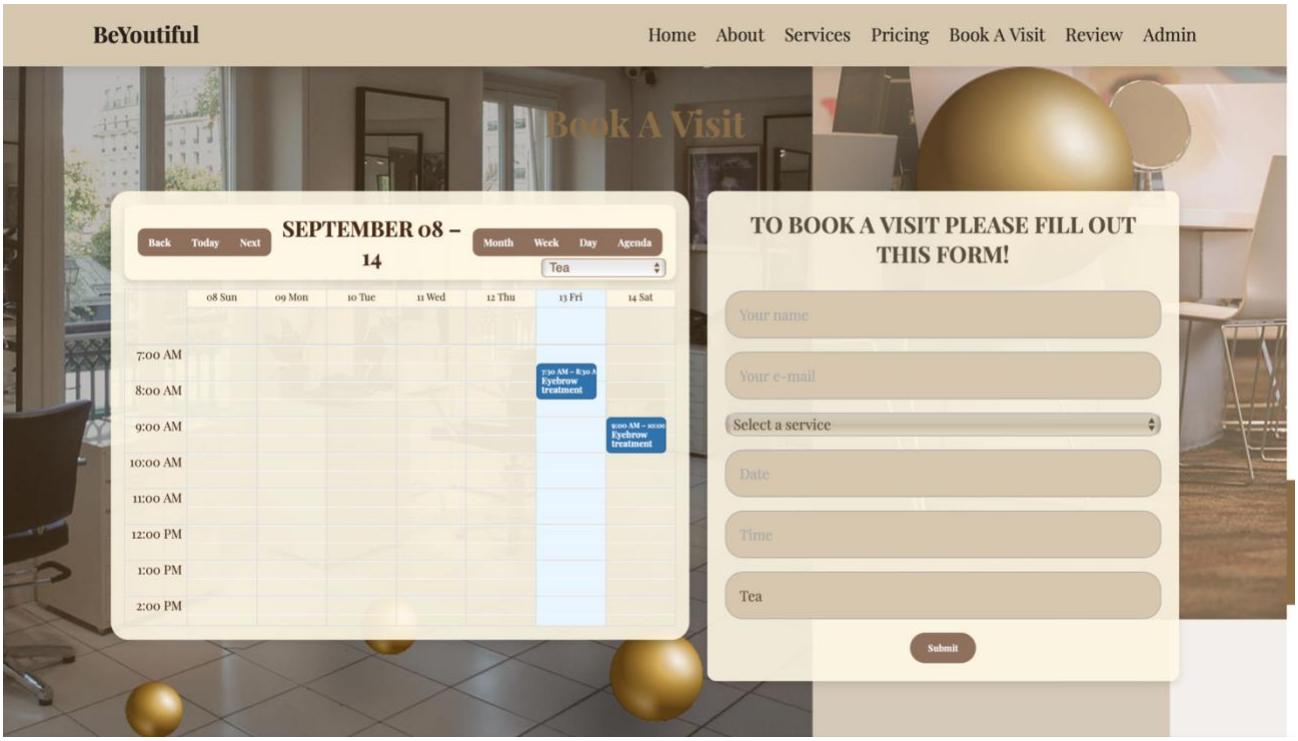
Slika 5.6. Prikaz kalendara drugog zaposlenika



Slika 5.7. Ispunjena forma nakon odabira datuma i vremena



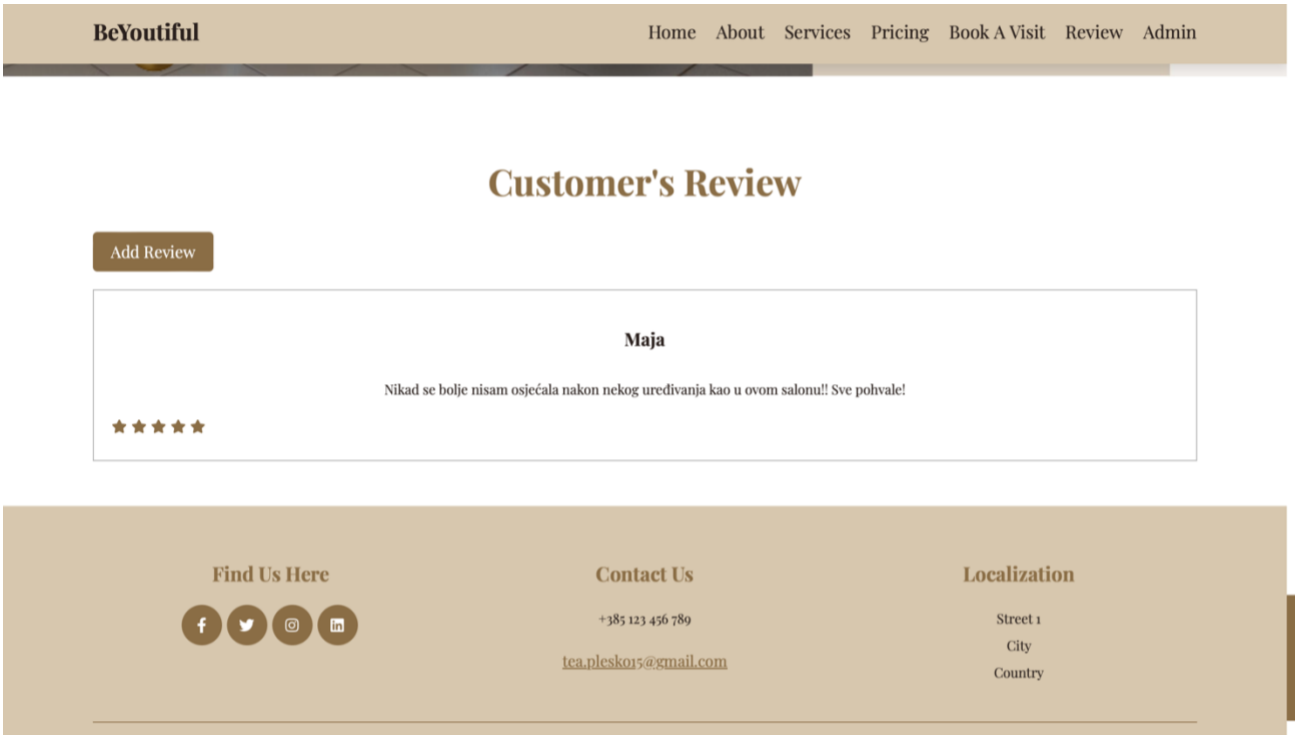
Slika 5.8. Poruka koja prikazuje uspješnost rezervacije termina



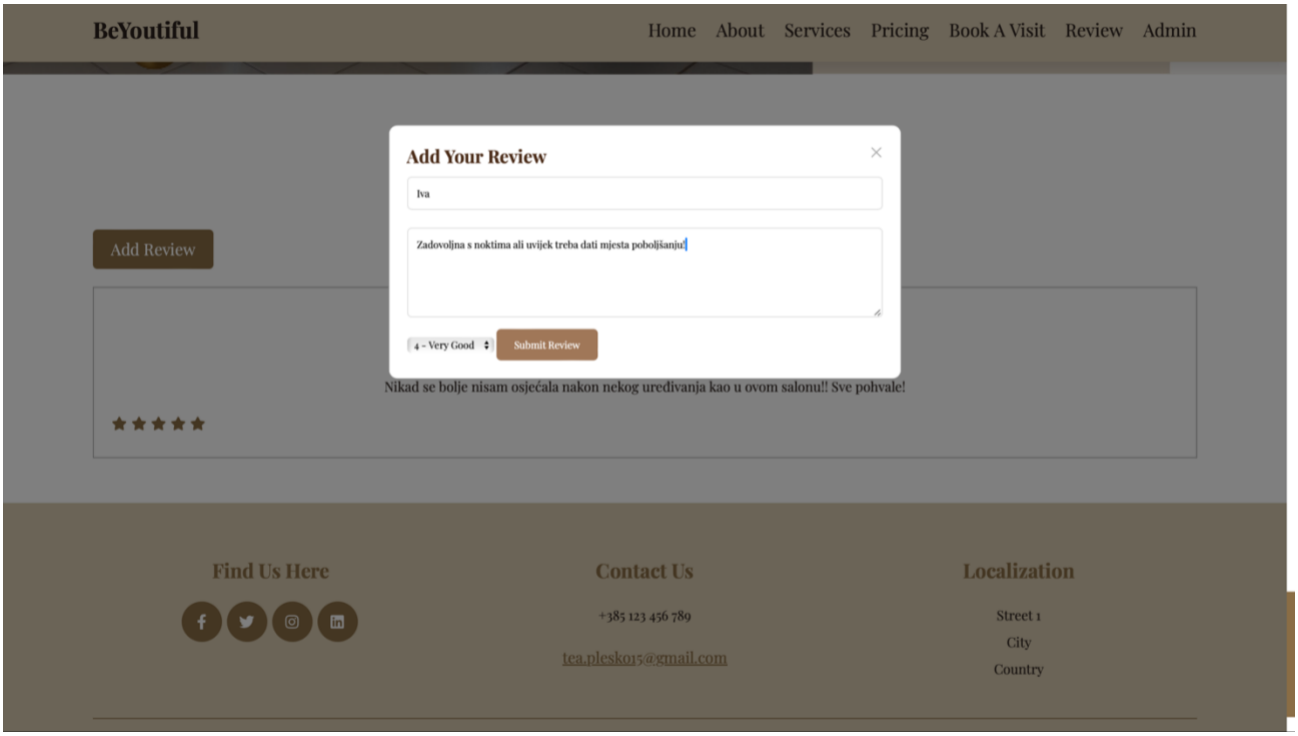
Slika 5.9. Novi rezervirani termin se prikazuje u kalendaru

5.3. Pisanje recenzije

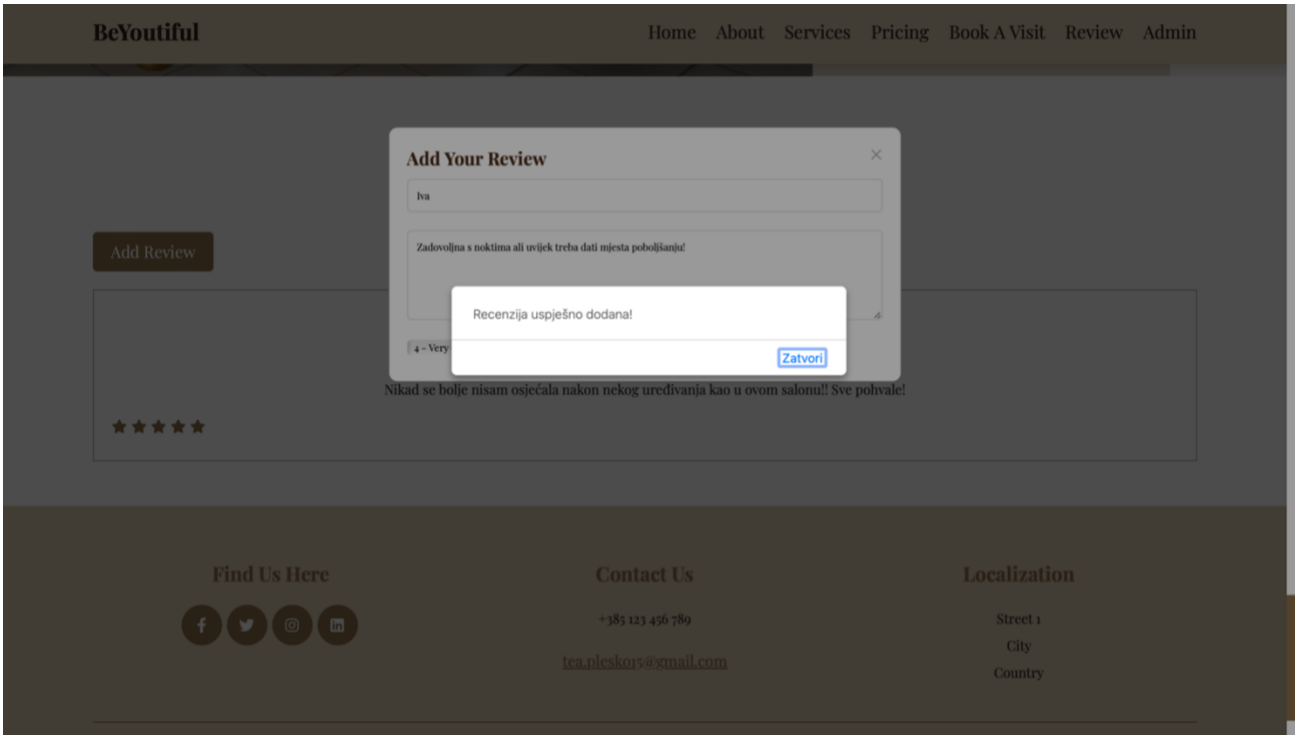
Sekcija za pisanje recenzije (Slika 5.10.) prikazuje trenutno jednu recenziju (autora, ocjenu i tekst recenzije). Pritiskom na gumb „Add Review“ korisniku se otvara modal (Slika 5.11.) koji sadržava formu za ostavljanje recenzije. Korisnik unese svoje ime, recenziju i odabere koju ocjenu želi pridružiti recenziji. Nakon što se pritisne gumb „Submit Review“ pojavljuje se poruka o uspješnosti slanja recenzije (Slika 5.12.) i u odjeljku recenzija se prikazuje najnovija recenzija (Slika 5.13.). Ocjena dodjeljena recenziji se prikazuje zvjezdicama, gdje je broj ocjene (1-5) jednak broju zvjezdica. U odjeljku je moguće prikazati 5 zadnjih recenzija što pruža korisnicima uvid u najnovija mišljenja i recenzije korisnika. Samo admin može vidjeti sve recenzije.



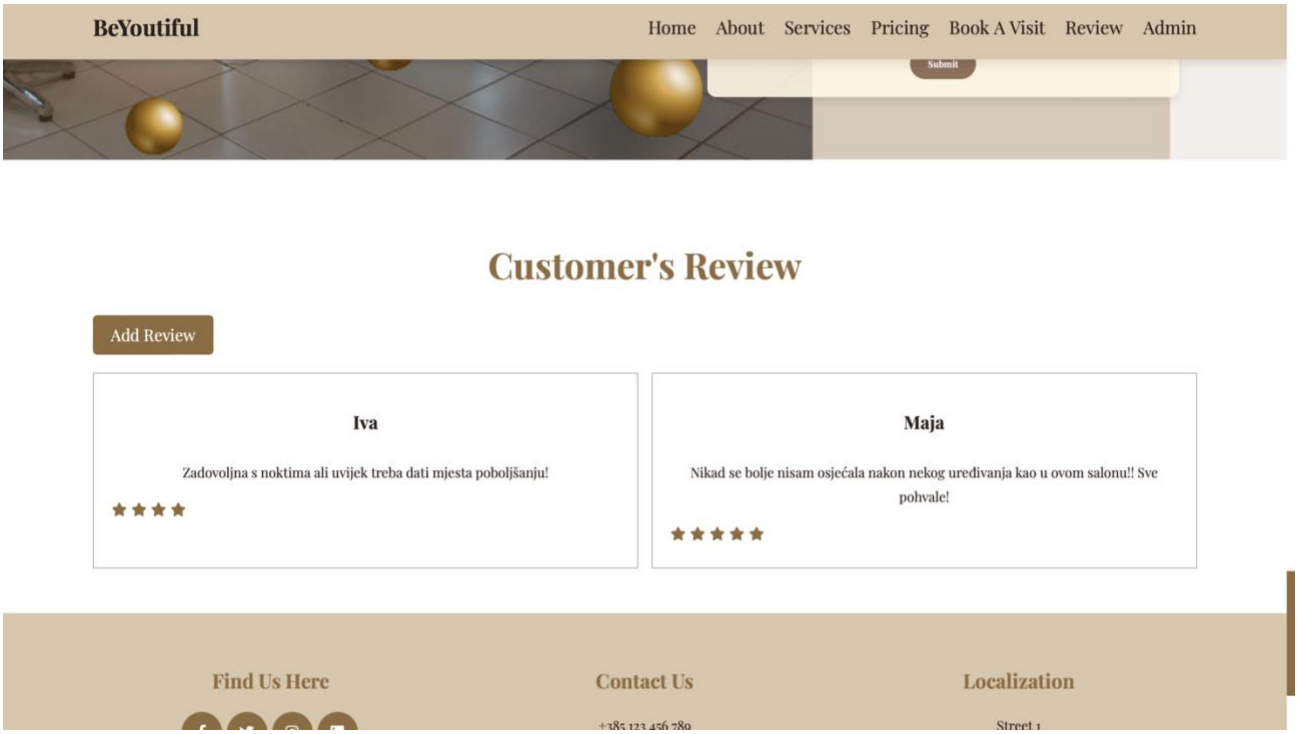
Slika 5.10. Odjeljak recenzija



Slika 5.11. Modal za unos nove recenzije



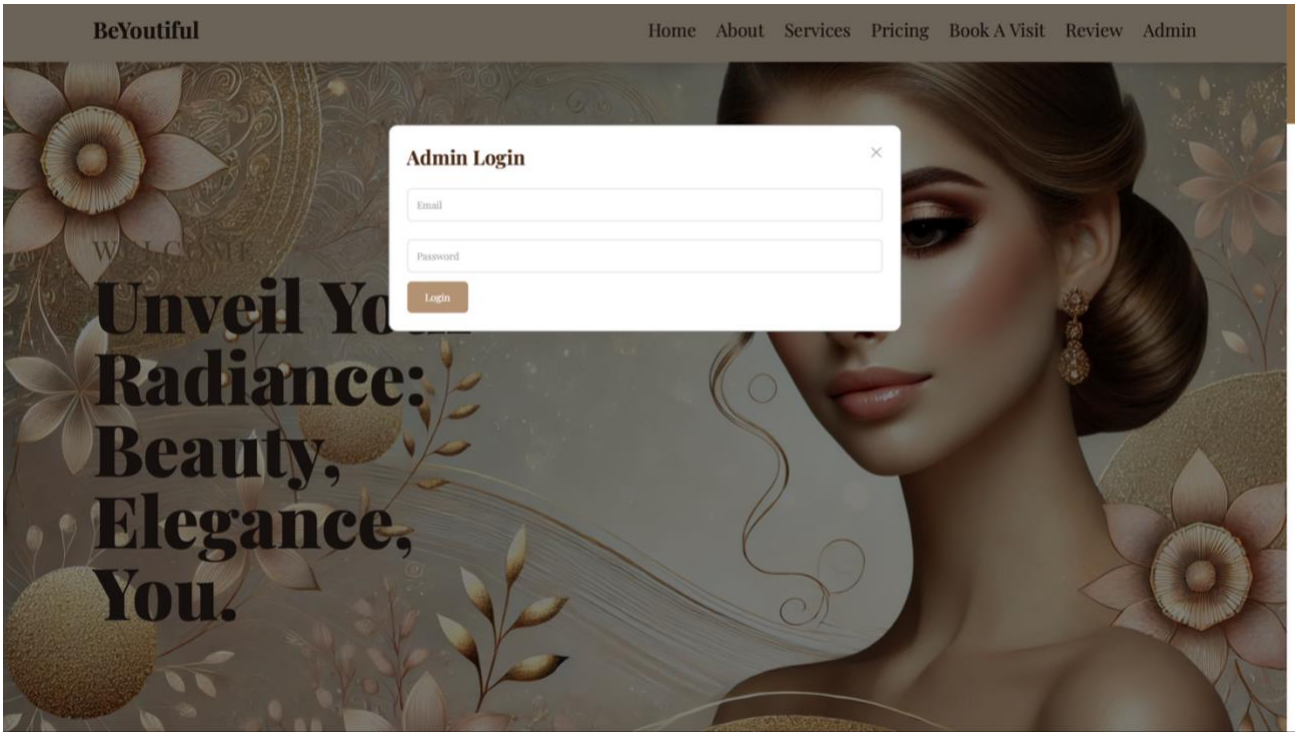
Slika 5.12. Poruka o uspješnosti dodavanja recenzije



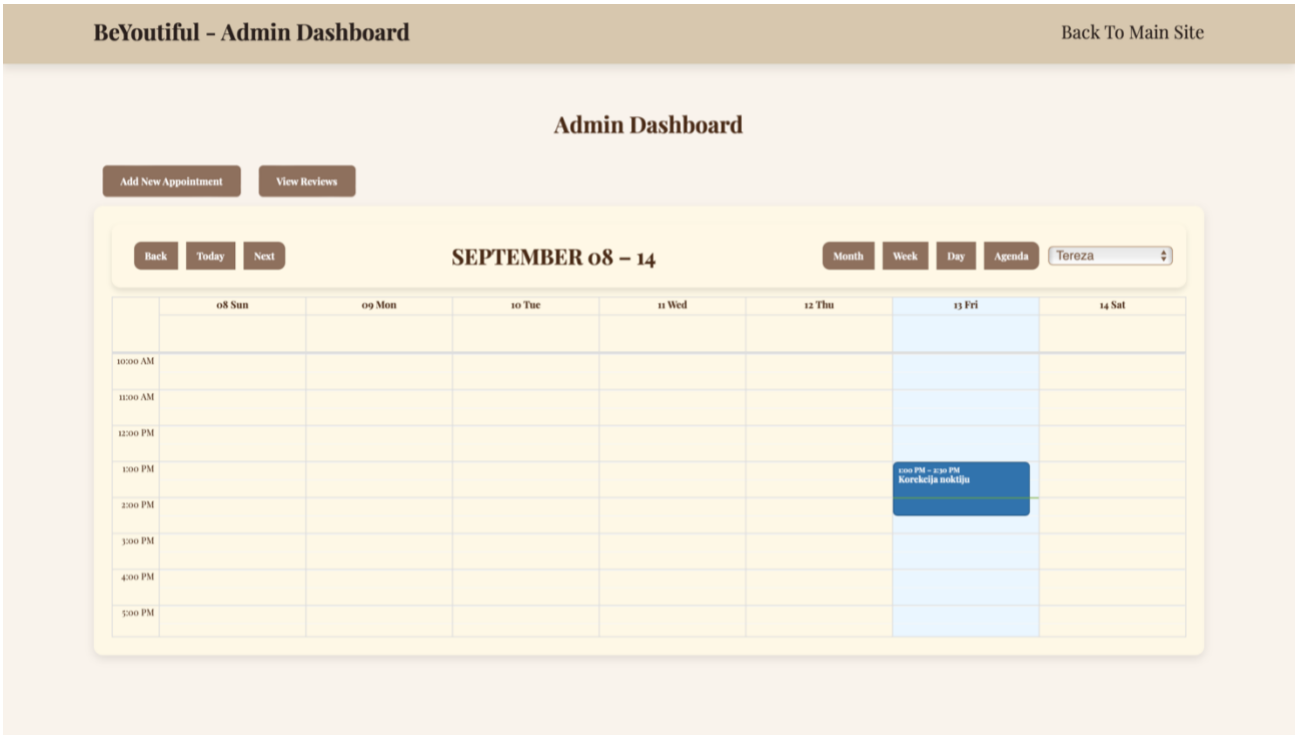
Slika 5.13. Ažurirana sekcija recenzija nakon dodavanja nove

5.4. Admin sučelje

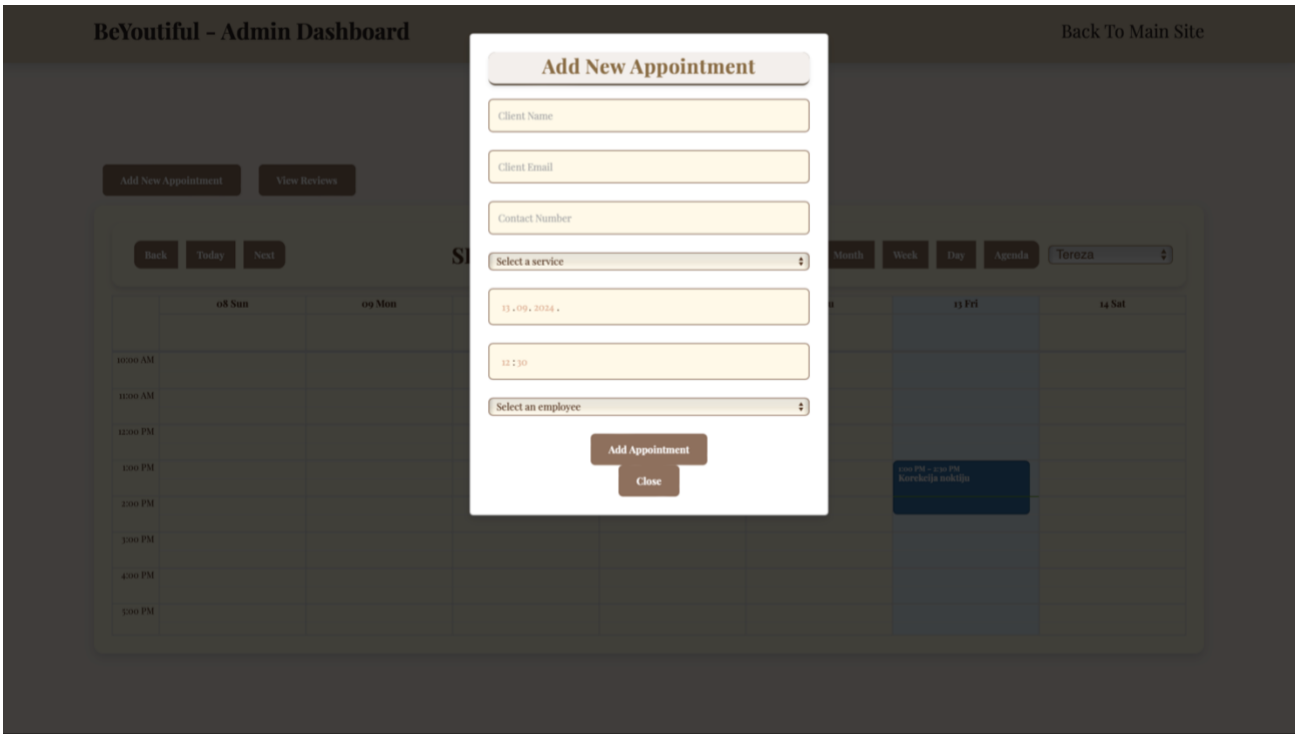
Kako bi zaposlenici mogli imati uvid u sve rezervacije, detalje, brisati ih i dodavati nove ili provjeriti kakvo mišljenje o salonu imaju korisnici putem ostavljenih recenzija, pritiskom na „Admin“ u navigaciji, otvara se modal (Slika 5.14.) koji omogućuje prijavu u sučelje namijenjeno njima. Nakon uspješnog unosa podataka za prijavu, otvara se sučelje (Slika 5.15.). Sadržaji sučelja su: kalendar na kojem je također moguće izabrati zaposlenika čiji raspored želimo vidjeti, gumb za dodavanje nove rezervacije termina i gumb za prikaz svih recenzija. Na kalendaru također možemo primijetiti kako postoji mala zelena linija koja označava trenutno vrijeme i možemo vidjeti koji događaj je u tijeku. Ako želimo dodati novi termin, otvara se modal (Slika 5.16.) u kojem je potrebno ispuniti polja kao i za korisničku rezervaciju. Nakon ispune polja, možemo vidjeti da je termin uspješno dodan i ažuriran u kalendar. Pritiskom na taj događaj, zaposlenici mogu vidjeti detalje svake rezervacije i nudi se mogućnost brisanja iste (Slika 5.17.). Zadnja funkcionalnost koju je potrebno prikazati su sve recenzije, čiji modal je vidljiv na slici 5.18.



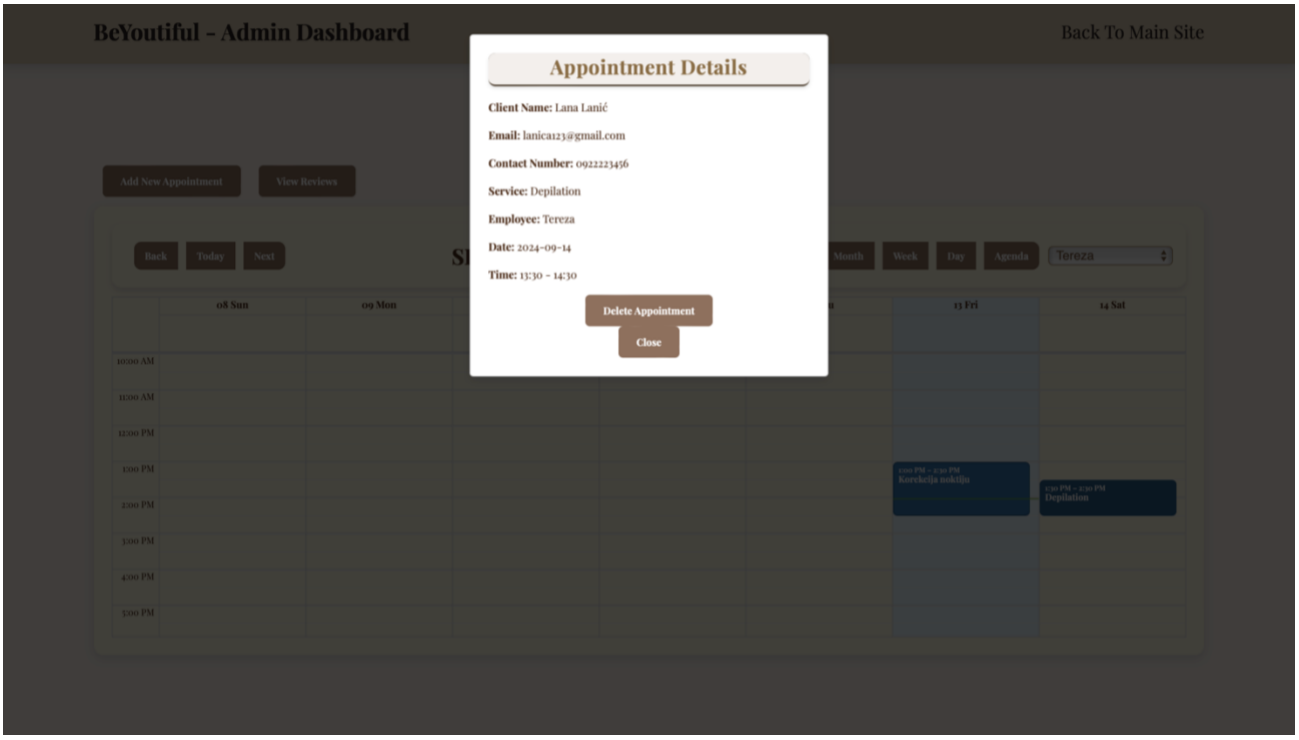
Slika 5.14. Modal za prijavu u admin sučelje



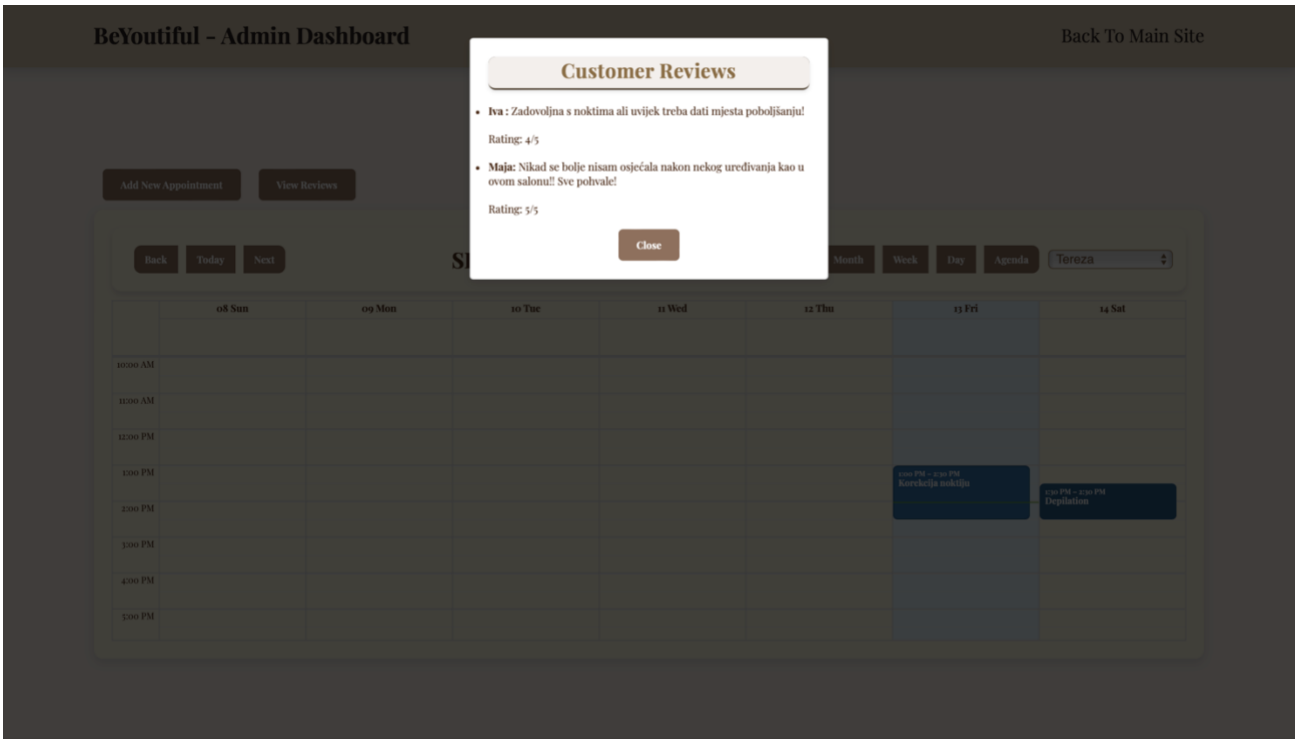
Slika 5.15. Admin sučelje



Slika 5.16. Modal za dodavanje nove rezervacije



Slika 5.17. Modal za prikaz detalja o rezervaciji



Slika 5.18. Modal za prikaz svih recenzija

6. ZAKLJUČAK

Aplikacija ostvaruje svoj glavni cilj: omogućava učinkovitu organizaciju rada kozmetičkog salona kroz sustav za upravljanje terminima i rasporedima rada zaposlenika. Korištenje modernih tehnologija poput Reacta za izradu dinamičkog korisničkog sučelja i Firebase platforme za pozadinske usluge osigurava brzinu, responzivnost i pouzdanost aplikacije. To je još važnije jer aplikacija upravlja velikom količinom podataka u stvarnom vremenu, poput rezervacija.

Uspješno je implementirana jedna od glavnih značajki - kalendar termina koji prikazuje raspoložive termine zaposlenika i obrazac za rezervaciju. Funkcija dohvaćanja podataka i sigurnosne kopije u stvarnom vremenu putem Firebasea osigurava da svi korisnici aplikacije odmah budu obaviješteni o promjenama, što je posebno važno kod vođenja kozmetičkog salona u kojem se svaki slobodan termin računa. Osim toga, dodana je mogućnost pisanja recenzija, što korisnicima omogućuje da izraze svoje mišljenje o usluzi. Ovaj element aplikaciji dodaje društveni aspekt, što može doprinijeti boljem korisničkom iskustvu i boljoj interakciji između salona i kupaca. Osim toga, nadzorna ploča administratora omogućuje vlasnicima salona ili zaposlenicima pregled svih termina u stvarnom vremenu, upravljanje njima i pregled komentara korisnika.

Sučelje je intuitivno, a implementacija React hookova omogućuje dinamičku manipulaciju podacima bez potrebe za osvježavanjem stranice, što doprinosi brzini i fluidnosti aplikacije. Korisnici svih dobnih skupina mogu jednostavno upravljati aplikacijom zahvaljujući intuitivnim navigacijskim elementima i jednostavnom vizualnom prikazu. Važno je spomenuti i uspješnu implementaciju responzivnog dizajna, koji osigurava podjednako dobar rad aplikacije na različitim uređajima, bilo da se radi o računalima, tabletima ili pametnim telefonima. Ovo je bitno u današnjem mobilnom svijetu, gdje mnogi korisnici pristupaju aplikacijama putem svojih mobilnih uređaja.

Trenutno aplikacija omogućuje administraciju preko jedne administratorske ploče, ali moglo bi se omogućiti više razina korisničkih prava. Na primjer, vlasnici salona mogli bi imati potpunu kontrolu nad aplikacijom, dok zaposlenici imaju ograničen pristup. Još jedna funkcionalnost koja bi mogla biti dodana jesu automatski podsjetnici i obavijesti. Implementacija značajke za automatsko slanje e-pošte ili SMS podsjetnika korisnicima za nadolazeće rezervacije mogla bi smanjiti broj propuštenih termina i povećati profesionalnost salona jer će korisnici pravovremeno dobivati obavijesti o svojim rezervacijama. No, trenutna izrađena web aplikacija za organizaciju rada

kozmetičkog salona predstavlja funkcionalno i korisnički prilagođeno rješenje za organizaciju poslovanja u industriji ljepote.

LITERATURA

[1] MDN Web Docs, HTML basics, dostupno na: https://developer.mozilla.org/en/US/docs/Learn/Getting_started_with_the_web/HTML_basics [1.9.2024.]

[2] W3Schools, HTML Introduction, dostupno na: https://www.w3schools.com/html/html_intro.asp [15.9.2024.]

[3] W3Schools, CSS Introduction, dostupno na: https://www.w3schools.com/css/css_intro.asp [15.9.2024.]

[4] GeeksforGeeks, CSS Introduction, dostupno na: https://www.geeksforgeeks.org/css-introduction/ [1.9.2024.]

[5] Mojwebdizajn.net, Uvod u JavaScript: JavaScript Introduction, dostupno na: https://www.mojwebdizajn.net/hr-hr/web-programiranje/uvod-u-javascript/uvod-u-javascript.php [1.9.2024.]

[6] D. Stančer, D. Horvatović, Osnove JavaScripta, priručnik za polaznike, Sveučilište u Zagrebu, Zagreb, 2019

[7] programiranje.com.hr, React, dostupno na: https://programiranje.com.hr/React [1.9.2024.]

[8] C. Desphande, The Best Guide to Know What Is React, srpanj 2024., dostupno na: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs [15.9.2024.]

[9] Google Help, O Firebaseu i AdMobu, dostupno na: https://support.google.com/admob/answer/6360054?hl=hr [1.9.2024.]

SAŽETAK

Ovaj rad opisuje razvoj web aplikacije za upravljanje radom kozmetičkog salona, s ciljem modernizacije poslovanja i poboljšanja korisničkog iskustva. Tradicionalne metode, poput ručnog bilježenja termina, postale su nepraktične, pa aplikacija omogućuje učinkovitije upravljanje rasporedom i rezervacijama. Web aplikacija je implementirana korištenjem tehnologija kao što su React za dinamično korisničko sučelje i Firebase za upravljanje podacima te autentifikaciju. Glavna funkcionalnost je sustav za online rezervaciju termina, koji se ažurira u stvarnom vremenu, omogućujući klijentima jednostavno pregledavanje, promjene i otkazivanja termin te recenziranje usluga. Administrativno sučelje omogućuje zaposlenicima upravljanje terminima. Aplikacija je responzivna i prilagodljiva za buduće proširenje.

Ključne riječi: autentifikacija, Firebase, kozmetički salon, React, rezervacija

ABSTRACT

Title: Web application for organizing the work of a beauty salon

This paper describes the development of a web application for managing beauty salon operations, aimed at modernizing the business and improving customer experience. Traditional methods, such as manually recording appointments, have become impractical, so the application enables more efficient scheduling and booking management. The web application is built using technologies like React for a dynamic user interface and Firebase for data management and authentication. The main feature is an online appointment booking system that updates in real-time, allowing clients to easily view, modify, cancel appointments, and review services. The administrative interface allows employees to manage appointments. The application is responsive and adaptable for future expansion.

Keywords: authentication, booking, Firebase, React, salon