



Univerzitet u Bihaću  
Tehnički fakultet

Odsjek: Elektrotehnika  
Smjer: Računarstvo i informatika

---

## **Objektno orijentirane baze podataka**

Projektni zadatak:

**Tema: Frizerski Salon**

---

Profesor: Prof. dr. Admir Midžić  
Asistent: MA Zinaid Kapić

Student:  
Elvedin Mujagić, 1257

Bihać, februar 2026.

## Sažetak

Frizerski salon predstavlja važan dio uslužne djelatnosti koji se bavi njegom kose, oblikovanjem frizura te unapređenjem osobnog izgleda klijenata. U ovom seminarском radu obrađuje se pojam frizerskog salona, njegova uloga u društvu, osnovne djelatnosti koje obuhvata, kao i značaj profesionalnosti i higijenskih standarda u radu. Poseban naglasak stavljen je na odnos frizera i klijenta, važnost komunikacije, praćenje modnih trendova te korištenje kvalitetnih proizvoda i savremenih tehnika rada. Također se razmatra organizacija rada u frizerskom salonu, značaj edukacije zaposlenih i utjecaj frizerske djelatnosti na zadovoljstvo i samopouzdanje klijenata. Cilj rada je prikazati frizerski salon kao mjesto koje objedinjuje estetiku, stručnost i kvalitetnu uslugu.

## Summary

A hair salon is an important part of the service industry that deals with hair care, hairstyle design, and enhancing the personal appearance of clients. This seminar paper explores the concept of a hair salon, its role in society, the main services it provides, as well as the importance of professionalism and hygiene standards in the work process. Special attention is given to the relationship between the hairdresser and the client, effective communication, keeping up with fashion trends, and using high-quality products and modern techniques. The organization of work within the salon, employee education, and the impact of hairdressing services on client satisfaction and self-confidence are also discussed. The aim of this paper is to present the hair salon as a place that combines aesthetics, expertise, and quality service.

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>1</b>
<b>2</b>	<b>Modeliranje Sistema</b>	<b>2</b>
2.1	Opis Aplikacije . . . . .	2
2.2	UML Dijagram . . . . .	3
2.3	Klasni Dijagram . . . . .	4
2.4	Dijagrami slučajeva korištenja . . . . .	5
2.5	Sekvencijalni dijagrami . . . . .	6
<b>3</b>	<b>Implementacija</b>	<b>7</b>
3.1	Tehnologija koristita u aplikaciji . . . . .	7
3.2	Laravel . . . . .	7
3.3	PostgreSQL . . . . .	8
3.4	Objektno-relaciono mapiranje . . . . .	8
3.5	REST Api . . . . .	9
<b>4</b>	<b>Analiza rada aplikacije</b>	<b>10</b>
4.1	Opis slučajeva korištenja . . . . .	10
4.2	Testiranje API-a . . . . .	20
<b>5</b>	<b>Zaključak</b>	<b>22</b>

## Popis slika

1	Klasni dijagram web aplikacije za Frizerski salon . . . . .	4
2	Klasni dijagram web aplikacije za Frizerski salon . . . . .	5
3	Sekvencijalni dijagram za Klijenta . . . . .	6
4	Prijava . . . . .	10
5	Ne uspješna prijava . . . . .	11
6	Rezervacija termina . . . . .	13
7	Ne usješna rezervacija termina . . . . .	13
8	Registracija . . . . .	14
9	Novi Korisnik . . . . .	15
10	Prikaz svih termina frizera . . . . .	16
11	Dodavanje usluge . . . . .	17
12	Dodana usluga . . . . .	18
13	Brisanje usluge . . . . .	18
14	Dodavanje Frizera . . . . .	20
15	Brisanje Frizera . . . . .	21
16	Ispis Usluga . . . . .	21

# 1 Uvod

Razvijeni projekat predstavlja web aplikaciju za upravljanje terminima i uslugama u frizerskom salonu, sa ciljem poboljšanja korisničkog iskustva i olakšavanja organizacije svakodnevnog rada. U današnjem digitalnom okruženju, sve više usluga prelazi na online platforme koje omogućavaju brzo zakazivanje termina, jednostavan pregled dostupnih usluga i direktnu komunikaciju između klijenata i frizerskog osoblja. Upravo zbog toga postojala je potreba za razvojem aplikacije koja će biti jednostavna za korištenje, pregledna i sigurna. Aplikacija je izrađena koristeći Laravel PHP framework, koji omogućava kreiranje stabilnih, sigurnih i skalabilnih web rješenja. Laravel se bazira na MVC arhitekturi, što omogućava jasnu raspodjelu zadataka unutar aplikacije. Modeli upravljaju bazom podataka, kontroleri obrađuju poslovnu logiku, dok pogledi (views) služe za prikaz korisničkog interfejsa. Ovak pristup olakšava održavanje, nadogradnju i testiranje aplikacije. Kroz aplikaciju, korisnici mogu pregledavati raspoložive frizere i usluge, rezervirati termine i pratiti svoje zakazane termine. Klijenti imaju mogućnost kreiranja termina, dok Admini imaju mogućnost praćenja termina u realnom vremenu. Za pohranu podataka koristi se PostgreSQL. Aplikacija uključuje osnovne sigurnosne mehanizme, poput autentifikacije korisnika, zaštite ruta i validacije podataka, čime se osigurava da samo ovlašteni korisnici mogu pristupiti administrativnim funkcijama. Glavni cilj projekta je kreiranje funkcionalne, pregledne i sigurne web aplikacije koja olakšava zakazivanje termina i doprinosi efikasnijem radu frizerskog salona. Korištenjem savremenih tehnologija i principa razvoja softvera, aplikacija pruža stabilno i skalabilno rješenje koje se može dalje prilagođavati potrebama korisnika i poslovanja.

## 2 Modeliranje Sistema

Planiranje i modeliranje aplikacije predstavlja osnovu za organizovan razvoj softverskog sistema, jer omogućava definisanje strukture, funkcionalnosti i međusobnih veza između različitih dijelova sistema. Prije početka implementacije potrebno je sagledati potrebe korisnika, odrediti ključne funkcionalnosti i osmisлити način na koji će korisnici koristiti aplikaciju. Takav pristup doprinosi efikasnijem razvoju, lakšem održavanju koda i smanjuje mogućnost pojave grešaka tokom razvoja. U projektu Frizerski Salon, modeliranje aplikacije obuhvata izradu dijagrama koji prikazuju tokove podataka, komunikaciju između korisnika i sistema, kao i strukturu baze podataka. Primijenjeni su dijagrami slučaja upotrebe (Use Case), dijagrami klasa i sekvencijski dijagrami, koji zajedno pružaju jasan pregled funkcionisanja aplikacije. Dijagrami ilustriraju ključne funkcionalnosti, uključujući registraciju i prijavu korisnika, pregled frizera i dostupnih usluga, rezervaciju termina, te upravljanje korisničkim profilima i zakazanim posjetama. Modeliranje omogućava bolje planiranje razvoja aplikacije, olakšava implementaciju i smanjuje rizik od grešaka, a istovremeno omogućava jednostavno dodavanje novih funkcionalnosti u budućnosti. Također, dokumentacija nastala tokom modeliranja služi kao vrijedan izvor informacija za sve članove razvojnog tima, olakšavajući razumijevanje strukture i logike sistema. Na ovaj način postavljeni su temelji za izgradnju stabilne, funkcionalne i skalabilne web aplikacije koja zadovoljava potrebe korisnika i osoblja frizerskog salona.

### 2.1 Opis Aplikacije

Frizerski Salon dizajnirana je kao web platforma koja korisnicima omogućava jednostavno zakazivanje termina i pristup informacijama o uslugama frizerskog salona. Primarni cilj aplikacije je pojednostaviti proces rezervacije, olakšati organizaciju rada frizerima i omogućiti klijentima pregled ponude i dostupnih termina na brz i pregledan način. Korisnici koji nisu registrovani mogu pristupiti osnovnim informacijama o salonu, raspoloživim frizerima i vrstama usluga, dok registrovani korisnici imaju dodatne mogućnosti, uključujući kreiranje ličnog profila, rezervaciju termina i praćenje svojih zakazanih posjeta. Svaka rezervacija sadrži podatke o korisniku, izabranoj usluzi i terminu, što omogućava efikasno praćenje i organizaciju radnog rasporeda. Frizeri i administratori imaju pristup alatima za upravljanje terminima, uključujući kreiranje novih termina, izmjenu postojećih i praćenje zakazanih posjeta, čime se olakšava koordinacija i održavanje reda u salonu. Aplikacija je kreirana tako da pruža sigurno i pregledno okruženje, sa fokusom na intuitivan korisnički interfejs i responzivan dizajn, omogućavajući kvalitetno iskustvo korištenja na različitim uređajima.

## 2.2 UML Dijagram

UML dijagrami služe za vizualno prikazivanje strukture softverskog sistema i njegovih osnovnih komponenti. Primarna svrha ovih dijagrama je olakšati razumijevanje dizajna aplikacije prije same implementacije, te pojednostaviti održavanje i dalji razvoj sistema. U procesu razvoja softverskih rješenja, UML dijagrami omogućavaju programerima i dizajnerima da jasno prikažu strukturu sistema, međusobnu povezanost komponenti i tokove podataka. Statički dijagrami fokusiraju se na prikaz trajnih elemenata sistema, bez prikazivanja njihovog ponašanja tokom izvršavanja. Postoje više vrsti dijagrama od kojih su najbitniji:

- Dijagram strukture sistema – prikazuje osnovne dijelove aplikacije i način na koji su povezani.
- Klasni dijagram – definiše klase, njihove attribute, metode i međusobne veze.
- Objektni dijagram – prikazuje konkretne instance klasa u određenom trenutku izvršavanja sistema.

Oni se koriste kako bi se opisala arhitektura aplikacije, međusobni odnosi elemenata i organizacija podataka unutar sistema.



## 2.3 Klasni Dijagram

Struktura aplikacije za frizerski salon može se jasno sagledati kroz klasni dijagram koji definiše najvažnije elemente sistema i veze između njih. Aplikacija sadrži klase Users, Services i Appointments, Work-Hours koje zajedno omogućavaju realizaciju osnovnih funkcionalnosti salona. Klijentima je omogućeno zakazivanje više termina, pri čemu je svaki termin povezan sa jednim frizerom i odabranom uslugom. Istovremeno, frizeri mogu imati više zakazanih termina, što omogućava efikasno planiranje radnog vremena. Ovakav model doprinosi boljoj organizaciji podataka, jednostavnijem upravljanju sistemom i predstavlja stabilnu osnovu za dalji razvoj i nadogradnju aplikacije.

Klasni dijagram aplikacije prikazan je na slici 1.

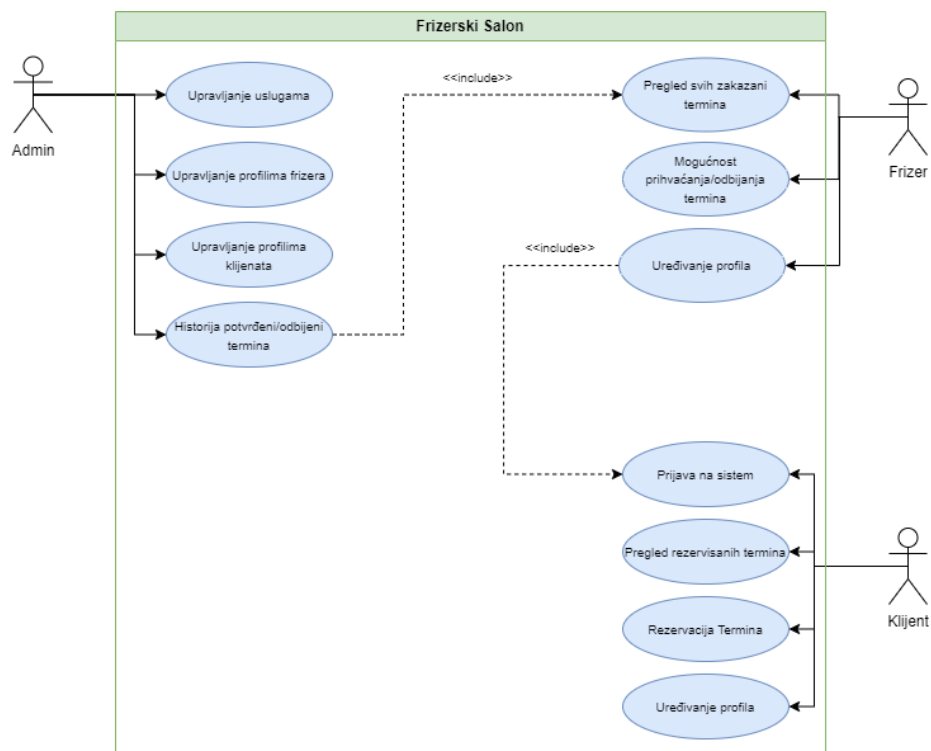


Slika 1: Klasni dijagram web aplikacije za Frizerski salon

## 2.4 Dijagrami slučajeva korištenja

Dijagrami slučajeva korištenja (Use Case dijagrami) služe za opis funkcionalnosti softverskog sistema iz ugla korisnika i načina na koji oni komuniciraju s aplikacijom. U okviru aplikacije za frizerski salon, ovi dijagrami omogućavaju jasan prikaz interakcije različitih vrsta korisnika sa sistemom, kao i radnji koje im stoje na raspolaganju. Fokus je stavljen na prikaz funkcionalnih zahtjeva sistema, bez razmatranja tehničkih detalja implementacije. Use Case dijagrami pomažu u definisanju tipičnih scenarija korištenja aplikacije, uloga klijenta i granica samog sistema, čime se olakšava razumijevanje načina na koji aplikacija funkcioniše u svakodnevnom radu frizerskog salona. Na ovaj način moguće je jasno sagledati ponašanje sistema u realnim uslovima i ulogu svakog aktera u procesu zakazivanja i upravljanja uslugama. U aplikaciji su definisani sljedeći akteri: Klijent, Frizer, Admin.

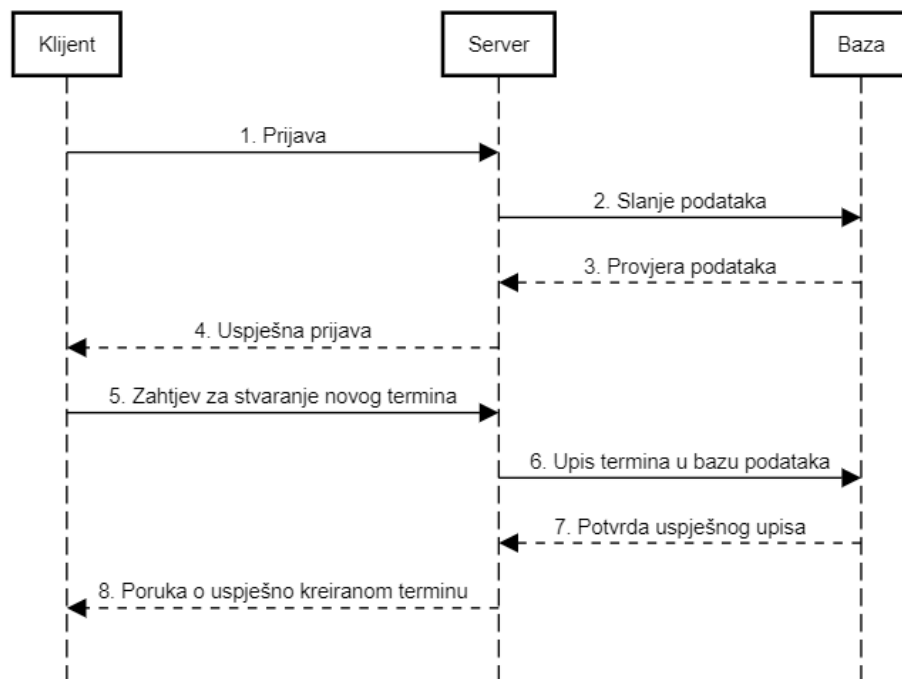
Prikaz slučajeva korištenja se nalazu sljedećoj slici



Slika 2: Klasni dijagram web aplikacije za Frizerski salon

## 2.5 Sekvencijalni dijagrami

Sekvencijalni dijagrami predstavljaju vrstu UML dijagrama koji se koriste za prikaz dinamike sistema, odnosno načina na koji objekti međusobno komuniciraju tokom izvršavanja određene funkcionalnosti. Njihova osnovna svrha je da prikažu redoslijed poruka i interakcija između pojedinih komponenti sistema u okviru određenog scenarija. Na ovaj način se jasno može pratiti tok izvršavanja procesa od početka do kraja. U kontekstu softverske aplikacije, sekvencijalni dijagrami omogućavaju bolje razumijevanje kako sistem reaguje na korisničke zahtjeve i kako se podaci razmjenjuju između različitih dijelova aplikacije. Svaki dijagram obično prikazuje učesnike u procesu (objekte ili klase) kao i poruke koje se razmjenjuju u tačno definisanom vremenskom slijedu. Ovi dijagrami pomažu u identifikaciji odgovornosti pojedinih komponenti sistema, smanjuju mogućnost grešaka tokom implementacije i olakšavaju razvoj funkcionalne i pouzdane aplikacije. Također, predstavljaju važan dio tehničke dokumentacije, jer pružaju jasan uvid u logiku rada sistema i njegovu interakciju sa korisnicima.



Slika 3: Sekvencijalni dijagram za Klijenta

## 3 Implementacija

### 3.1 Tehnologija koristita u aplikaciji

Razvoj aplikacije realizovan je korištenjem programskog jezika PHP uz upotrebu Laravel frameworka, koji se široko koristi za izradu savremenih web rješenja. Laravel je zasnovan na MVC (Model–View–Controller) arhitekturnom obrascu, koji omogućava jasnu podjelu odgovornosti unutar aplikacije, odvajajući poslovnu logiku, upravljanje podacima i korisnički interfejs. Ovakav pristup doprinosi boljoj organizaciji koda, jednostavnijem održavanju i većoj preglednosti kompletnog sistema. U okviru MVC arhitekture, modeli su zaduženi za obradu podataka i komunikaciju s bazom podataka, kontroleri obrađuju zahtjeve korisnika i upravljaju poslovnom logikom, dok su prikazi odgovorni za vizuelni prikaz informacija krajnjim korisnicima. Ova struktura omogućava lakše testiranje aplikacije i jednostavno proširenje postojećih funkcionalnosti. Za pohranu i upravljanje podacima koristi se PostgreSQL sistem za upravljanje relacionim bazama podataka, koji pruža visoku pouzdanost, sigurnost i efikasnost u radu sa složenim skupovima podataka. PostgreSQL je posebno pogodan za aplikacije koje upravljaju većim brojem međusobno povezanih entiteta, kao što su klijenti, frizeri, usluge i termini u frizerskom salonu. Kombinacija Laravel frameworka i PostgreSQL baze podataka obezbjeđuje stabilnu, skalabilnu i robusnu osnovu za razvoj funkcionalne web aplikacije namijenjene radu frizerskog salona.

### 3.2 Laravel

Laravel je moderan PHP framework koji se koristi za razvoj web aplikacija na brz, organizovan i siguran način. Predstavlja skup alata i pravila koji programerima olakšavaju izradu funkcionalnih i skalabilnih sistema, bez potrebe za pisanjem velike količine ponavljajućeg koda. Laravel je posebno popularan zbog svoje čiste strukture, čitljivog koda i bogate dokumentacije. Osnovu Laravela čini MVC (Model–View–Controller) arhitektura, koja omogućava jasno razdvajanje poslovne logike, rada s podacima i korisničkog interfejsa. Na taj način se postiže bolja organizacija projekta, lakše održavanje aplikacije i jednostavnija nadogradnja funkcionalnosti. Laravel pruža ugrađene mehanizme za rutiranje, validaciju podataka, autentifikaciju korisnika i upravljanje sesijama. Jedna od važnih karakteristika Laravela je Eloquent ORM, koji omogućava rad s bazom podataka kroz objektno orijentisan pristup, bez potrebe za pisanjem složenih SQL upita. Također, Laravel nudi snažne sigurnosne mehanizme, poput zaštite od SQL injection napada, cross-site scriptinga (XSS) i cross-site request forgery (CSRF). Zahvaljujući svojoj fleksibilnosti i velikom broju dodatnih alata, Laravel se često koristi za razvoj različitih web aplikacija, uključujući sisteme za upravljanje sadržajem, web portale, kao i aplikacije za upravljanje poslovnim procesima, poput aplikacije za frizerski salon.

### 3.3 PostgreSQL

PostgreSQL je napredni sistem za upravljanje relacionim bazama podataka (DBMS) otvorenog koda, koji se koristi za pouzdano i sigurno čuvanje i obradu podataka u savremenim softverskim aplikacijama. Poznat je po svojoj stabilnosti, visokoj efikasnosti i snažnoj podršci za složene upite i velike količine podataka. PostgreSQL omogućava rad sa relacionim bazama podataka kroz korištenje standardnog SQL jezika, uz dodatnu podršku za napredne funkcionalnosti kao što su transakcije, integritet podataka, ograničenja i indeksiranje. Sistem pruža visok nivo sigurnosti, uključujući kontrolu pristupa, autentifikaciju korisnika i mehanizme za zaštitu podataka. Jedna od važnih prednosti PostgreSQL-a je njegova sposobnost rada sa složenim strukturama podataka i velikim brojem međusobno povezanih tabela. Zbog toga je posebno pogodan za aplikacije koje zahtijevaju pouzdano upravljanje informacijama, kao što su sistemi za zakazivanje termina, evidenciju klijenata i usluga u frizerskom salonu. Zahvaljujući svojoj fleksibilnosti, skalabilnosti i podršci za različite operativne sisteme, PostgreSQL se često koristi u profesionalnim i produkcijskim okruženjima, te predstavlja pouzdanu osnovu za razvoj stabilnih i dugotrajnih softverskih rješenja.

### 3.4 Objektno-relaciono mapiranje

Objektno-relaciono mapiranje (ORM) predstavlja tehniku koja omogućava povezivanje objektno orijentisanog programiranja sa relacionim bazama podataka. Cilj ORM-a je da olakša rad s podacima tako što se zapisi iz baze predstavljaju kao objekti u aplikaciji, čime se smanjuje potreba za direktnim pisanjem SQL upita. Na ovaj način programeri mogu raditi s bazom podataka koristeći koncepte poznate iz objektno orijentisanog programiranja, poput klasa, objekata i metoda. Korištenjem ORM-a postiže se jasnija struktura koda i veća preglednost aplikacije, jer se logika za rad s podacima odvaja od ostatka sistema. ORM omogućava jednostavno kreiranje, čitanje, ažuriranje i brisanje podataka (CRUD operacije), kao i definisanje veza između entiteta, poput odnosa jedan-prema-više ili više-prema-više. Ovakav pristup je posebno koristan u aplikacijama koje upravljaju većim brojem povezanih podataka, kao što su sistemi za zakazivanje termina u frizerskom salonu. U kontekstu Laravel frameworka, ORM rješenje poznato kao Eloquent omogućava intuitivan i efikasan rad sa bazom podataka. Svaka tabela u bazi predstavlja se odgovarajućom klasom, dok redovi u tabeli odgovaraju instancama tih klasa. Eloquent ORM pojednostavljuje komunikaciju sa PostgreSQL bazom podataka i doprinosi bržem razvoju, lakšem održavanju i većoj pouzdanosti aplikacije.

### 3.5 REST Api

REST API (Representational State Transfer Application Programming Interface) je način komunikacije između različitih softverskih sistema putem interneta. Zasniva se na jasno definisanim pravilima koja omogućavaju razmjenu podataka između klijenta i servera na jednostavan, standardizovan i efikasan način. REST API se najčešće koristi u web i mobilnim aplikacijama za dohvat, slanje i obradu podataka. Osnovna ideja REST arhitekture je da se resursi sistema, poput korisnika, termina ili usluga u frizerskom salonu, predstavljaju putem jedinstvenih URL adresa. Nad tim resursima se izvršavaju operacije koristeći HTTP metode, kao što su GET za dohvat podataka, POST za kreiranje novih zapisa, PUT ili PATCH za ažuriranje postojećih podataka i DELETE za brisanje. Podaci se najčešće razmjenjuju u JSON formatu, koji je lagan i lako čitljiv. REST API omogućava razdvajanje korisničkog interfejsa od poslovne logike sistema, čime se postiže veća fleksibilnost i skalabilnost aplikacije. Na primjer, backend aplikacija za frizerski salon može pružati REST API koji koriste web i mobilne aplikacije za zakazivanje termina, pregled usluga i upravljanje korisničkim nalogima. Ovakav pristup olakšava održavanje sistema, omogućava lakšu integraciju sa drugim servisima i poboljšava ukupnu organizaciju softverskog rješenja.

## 4 Analiza rada aplikacije

Analiza rada aplikacije je izvršena kroz pregled osnovnih slučajeva korištenja i identifikovane moguće povratne akcije sistema.

### 4.1 Opis slučajeva korištenja

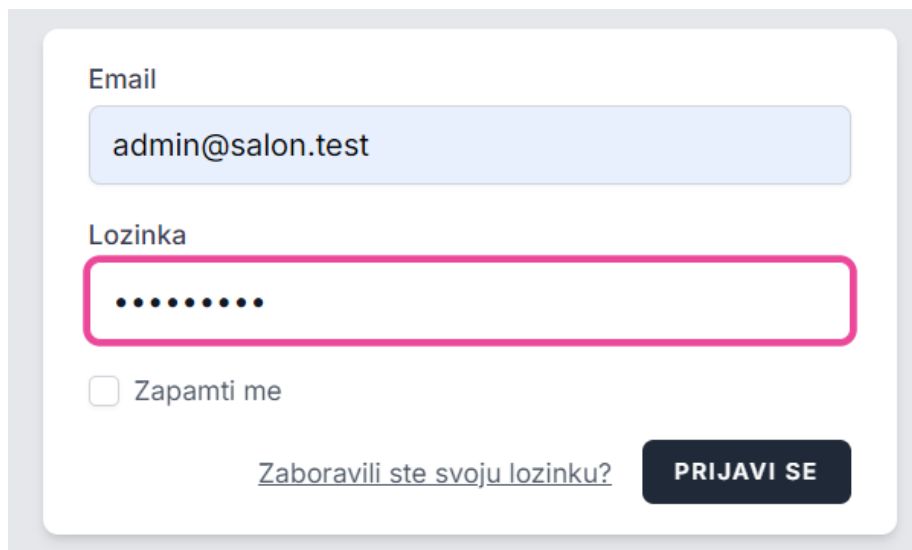
#### Slučaj korištenja 1: Prijava

Osnovni scenarij:

1. Korisnik unese ulazne podatke
2. Sistem provjeri da li su tačni podatci
3. Sistem dozvoli korisniku na sljedećem dijelu aplikacije

Sporedni scenarij:

1. Podatci nisu ispravni
2. Ispisuje se poruka za neispravne podatke



The image shows a login form with the following elements:

- Email:** A text input field containing the email address `admin@salon.test`.
- Lozinka:** A password input field with a pink border and masked characters (dots).
- Zapamti me:** A checkbox followed by the text "Zapamti me".
- Zaboravili ste svoju lozinku?:** A text link below the password field.
- PRIJAVI SE:** A dark blue button with white text.

Slika 4: Prijava

**Greške prilikom registracije!**

- These credentials do not match our records.

Email

admin@salon.test

Lozinka

☐ Zapamti me

[Zaboravili ste svoju lozinku?](#) **PRIJAVI SE**

Slika 5: Ne uspješna prijava



## **Slučaj korištenja 2: Zakazivanje termina**

Osnovni scenarij:

1. Klijent unese ulazne podatke
2. Sistem provjeri da li su tačni podatci
3. Sistem sprema termin u bazu podataka

Sporedni scenarij:

1. Podatci nisu ispravni
2. Ispisuje se poruka za neispravne podatke

Rezervacija je kreirana i čeka potvrdu frizera.

**Zatraži novi termin**

Stylist

Odaberite frizera

Termin mora biti unutar radnog vremena odabranog frizera.

Service

Odaberite uslugu

Datum i vrijeme

dd/mm/2026 --:--

Napomena

Napomene

Zakazi termin

Slika 6: Rezervacija termina

**Zatraži novi termin**

Stylist

Ana Kovač (08:00–16:00)

Termin mora biti unutar radnog vremena odabranog frizera.

Service

Brijanje - 10.00 KM - 15 Min

Datum i vrijeme

19/02/2026 02:15

Odabrano vrijeme nije u radnom vremenu frizera. Radno vrijeme ovog frizera je od 08:00 do 16:00.

Napomena

Napomene

Zakazi termin

Slika 7: Ne usješna rezervacija termina

### Slučaj korištenja 3: Registracija

Osnovni scenarij:

1. Korisnik unese ulazne podatke(email,password,ime,prezime...)
2. Sistem provjeri da li podatci validni
3. Sistem sprema korisnika kao klijenta u bazu podataka

The image shows a registration form with the following elements:

- Ime**: A text input field for the user's first name.
- Prezime**: A text input field for the user's last name.
- Email**: A text input field for the user's email address.
- Lozinka**: A text input field for the user's password.
- Potvrdi lozinku**: A text input field for the user to confirm their password.
- Već ste registrovani? Prijavite se!**: A link for existing users to log in.
- REGISTRIRAJ SE**: A pink button to initiate the registration process.

Slika 8: Registracija

[← Home](#) **Moji termini**

**Zatraži novi termin**

Stylist

Odaberite frizera

Termin mora biti unutar radnog vremena odabranog frizera.

Service

Odaberite uslugu

Datum i vrijeme

dd/mm/2026 --|--

Napomena

Napomene

Zakazi termin

**Moji termini**

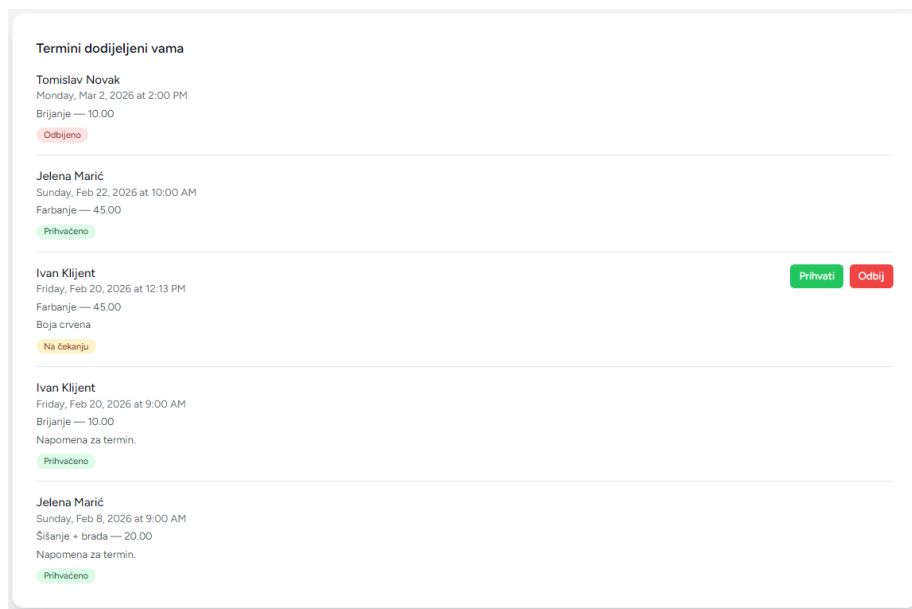
Nemate zakazanih termina.

Slika 9: Novi Korisnik

## Slučaj korištenja 4: Prihvaćanje i odbijanje termina

Osnovni scenarij:

1. Frizer odabere da li će prihvatiti/odbiti termin
2. Sistem sprema odluku u bazu podataka

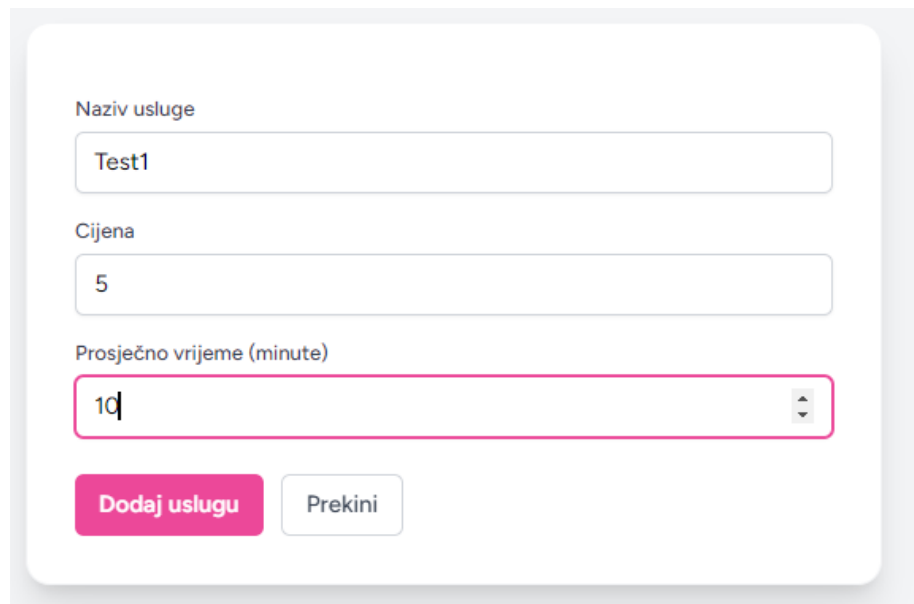


Slika 10: Prikaz svih termina frizera

### Slučaj korištenja 5: Dodavanje i brisanje usluge

Osnovni scenarij:

1. Admin unese ulazne podatke(Ime, Cijenu, Vrijeme)
2. Sistem sprema unešeni servis u bazu podataka
3. Sistem briše uslugu sa baze podataka



The image shows a web form for adding a service. It contains three input fields: 'Naziv usluge' with the value 'Test1', 'Cijena' with the value '5', and 'Prosječno vrijeme (minute)' with the value '10'. Below the fields are two buttons: a pink 'Dodaj uslugu' button and a grey 'Prekini' button.

Field Label	Value
Naziv usluge	Test1
Cijena	5
Prosječno vrijeme (minute)	10

Buttons: Dodaj uslugu, Prekini

Slika 11: Dodavanje usluge

Usluga je uspješno kreirana.

NAME	PRICE	AVG.TIME	ACTIONS	
Brijanje	10.00	15 min	Uredi	Obriši
Farbanje	45.00	1h 30min	Uredi	Obriši
Pramenovi	55.00	2h	Uredi	Obriši
Šišanje + brada	20.00	45 min	Uredi	Obriši
Šišanje muško	15.00	30 min	Uredi	Obriši
Šišanje žensko	25.00	1h	Uredi	Obriši
Test1	5.00	10 min	Uredi	Obriši

Slika 12: Dodana usluga

Usluga je uspješno obrisana.

NAME	PRICE	AVG.TIME	ACTIONS	
Brijanje	10.00	15 min	Uredi	Obriši
Farbanje	45.00	1h 30min	Uredi	Obriši
Pramenovi	55.00	2h	Uredi	Obriši
Šišanje + brada	20.00	45 min	Uredi	Obriši
Šišanje muško	15.00	30 min	Uredi	Obriši
Šišanje žensko	25.00	1h	Uredi	Obriši

Slika 13: Brisanje usluge

### **Slučaj korištenja 6: Dodavanje/Brisanje Frizera**

Osnovni scenarij:

1. Admin unese ulazne podatke(Ime, Prezime, Email, Password...)
2. Sistem sprema unešenog Frizera u bazu podataka
3. Sistem briše Frizera sa baze podataka



Kreirajte novi korisnički račun s ulogom frizera. Oni će moći da se prijave i upravljaju terminima koji su im dodijeljeni.

Ime

Prezime

Email

Lozinka

Potvrdi lozinku

Početak radnog vremena

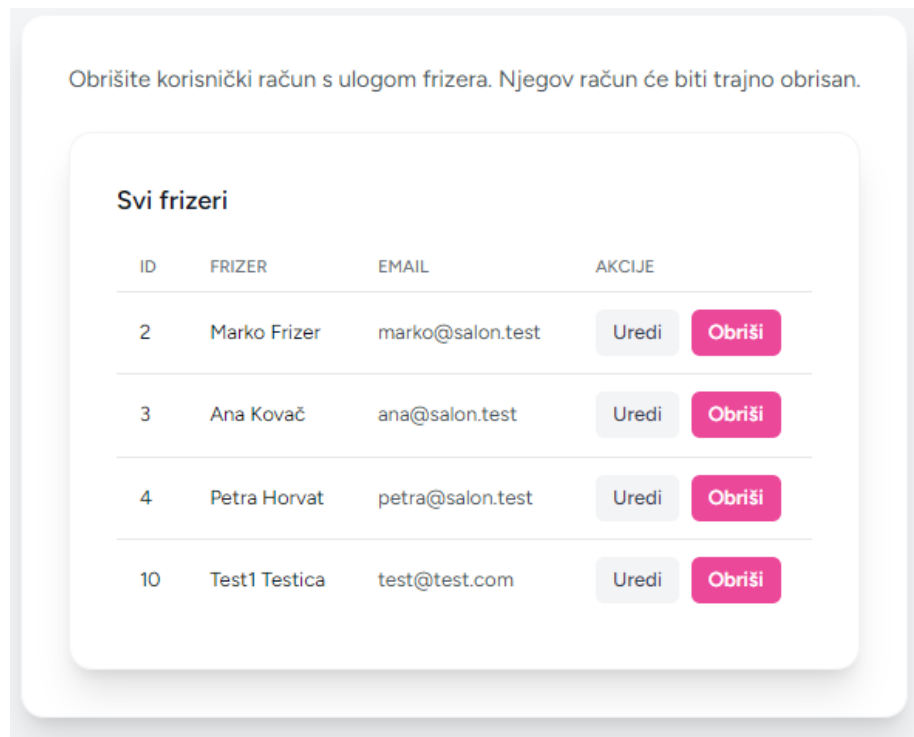
Kraj radnog vremena

**Napravi frizera** Prekini

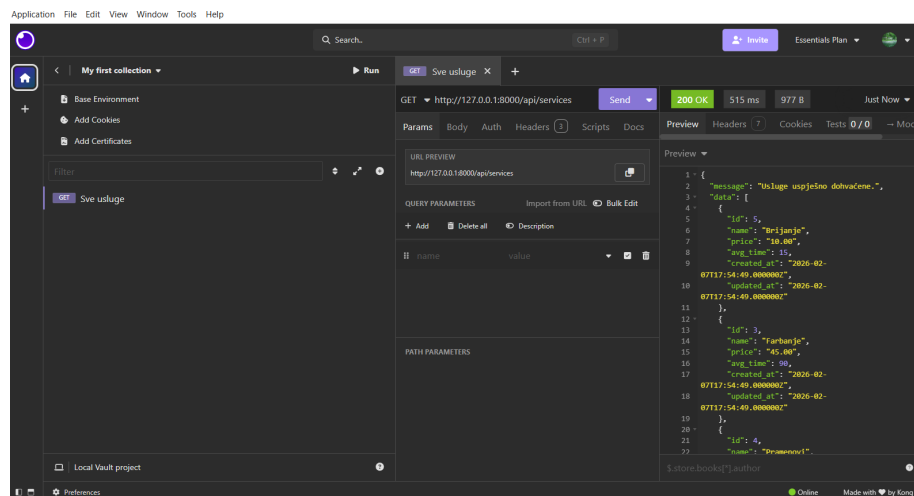
Slika 14: Dodavanje Frizera

## 4.2 Testiranje API-a

Za testiranje API-a korišten je alat Insomnia, koji omogućava jednostavno slanje HTTP zahtjeva i pregled API-ja u JSON formatu. Tokom testiranja API-ja praćeno je na ispravnost unosa podataka. Na slici je prikazan API za ispis svih usluga u aplikaciji.



Slika 15: Brisanje Frizera



Slika 16: Ispis Usluga

## 5 Zaključak

Razvijena web aplikacija za upravljanje frizerskim salonom predstavlja sveobuhvatno softversko rješenje koje olakšava svakodnevne poslovne procese, poboljšava organizaciju rada i omogućava bolju kontrolu nad terminima, uslugama i korisničkim nalogima. Za razvoj aplikacije korišten je programski jezik PHP u kombinaciji sa Laravel frameworkom, koji omogućava efikasno i strukturirano programiranje web aplikacija. Laravel je zasnovan na MVC (Model–View–Controller) arhitekturi, što omogućava jasno razdvajanje poslovne logike, prikaza podataka i upravljanja bazom. Modeli su zaduženi za rad s podacima i komunikaciju s bazom, kontroleri obrađuju korisničke zahtjeve i implementiraju poslovnu logiku, dok prikazi (views) pružaju pregledan i intuitivan interfejs korisnicima. Takav arhitektonski pristup doprinosi boljoj organizaciji koda, lakšem održavanju i jednostavnoj nadogradnji funkcionalnosti u budućnosti. Za pohranu podataka korišten je PostgreSQL, pouzdana i napredna relacionalna baza podataka koja omogućava efikasno rukovanje velikim brojem međusobno povezanih entiteta. PostgreSQL podržava napredne upite, integritet podataka i transakcije, što ga čini idealnim za aplikacije koje upravljaju kompleksnim podacima, poput termina, usluga, klijenata i frizera u salonu. Korištenjem Eloquent ORM-a, Laravel omogućava objektno-orijentisan pristup bazi podataka, što pojednostavljuje manipulaciju podacima i smanjuje potrebu za direktnim SQL upitima. Ovakav pristup doprinosi preglednosti koda, bržem razvoju i većoj pouzdanosti sistema. Primjena REST API-ja omogućava komunikaciju između frontend i backend dijela aplikacije, te omogućava jednostavnu integraciju sa mobilnim aplikacijama ili drugim servisima u budućnosti. Ova modularnost i fleksibilnost sistema doprinosi dugoročnoj održivosti i skalabilnosti rješenja. Kroz ovaj projekat demonstrirana je praktična primjena savremenih principa razvoja softvera, od planiranja i modeliranja, preko arhitekture i implementacije, do testiranja i validacije funkcionalnosti. Kombinacija Laravel frameworka, PostgreSQL baze podataka, Eloquent ORM-a, REST API-ja i detaljnog UML modeliranja rezultirala je stabilnom, sigurnom i funkcionalnom web aplikacijom, koja se može dalje nadograđivati i prilagođavati potrebama frizerskog salona.

## Literatura

- [1] Laravel, Dostupno na linku: <https://laravel.com/>
- [2] UML, Dostupno na linku: <https://developer.ibm.com/articles/an-introduction-to-uml/>
- [3] PostgreSQL, Dostupno na linku: <https://www.postgresql.org/about/>