



UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
ODSJEK: ELEKTROTEHNIKA
SMJER: RAČUNARSTVO I INFORMATIKA

Objektno orijentirane baze podataka

PROJEKTNI ZADATAK

TEMA: UPRAVLJANJE USLUGAMA AUTODETAILING
FIRME

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

Student:
Hadis Mahmutović, 1271

Bihać, februar 2026. godine

Sažetak

U savremenom razvoju informacionih sistema sve veći značaj ima izrada aplikacija koje omogućavaju efikasno upravljanje poslovnim procesima i pouzdanu evidenciju podataka. U ovom radu razvijena je web aplikacija za upravljanje uslugama autodetailing firme, koja omogućava digitalno praćenje termina, korisnika, proizvoda i poslovica. Aplikacija je razvijena korištenjem Laravel frameworka, uz Tailwind za responzivni korisnički interfejs, Livewire za interaktivne komponente i Jetstream za sigurnu autentifikaciju korisnika.

Sistem omogućava registraciju korisnika samostalno ili putem administratora, upravljanje korisnicima podijeljenim po različitim ulogama, kreiranje i pregled termina za različite vrste usluga, te odobravanje ili odbijanje termina od strane zaposlenika i poslovodje. Pored toga, aplikacija uključuje upravljanje web shopom proizvoda dostupnih u poslovcicama, upravljanje samim poslovcicama sa njihovim zaposlenicima i rasporedom termina, kao i pregled lokacija poslovica pomoću Leaflet API-ja. Podaci se pohranjuju u MySQL relacionu bazu podataka, dok je korisnički interfejs dizajniran da bude pregledan i prilagođen svakodnevnom radu osoblja. Također je implementiran REST API koji omogućava programski pristup podacima korisnika i integraciju sa vanjskim sistemima.

Aplikacija predstavlja praktično rješenje za digitalizaciju poslovanja autodetailing firme, olakšavajući upravljanje terminima, proizvodima i korisnicima kroz moderan, funkcionalan i siguran sistem.

Ključne riječi: web aplikacija, autodetailing, upravljanje terminima, Laravel, Livewire, Tailwind CSS, Jetstream, REST API, rezervacije usluga, upravljanje korisnicima, Leaflet API, baza podataka

Abstract

In modern information system development, increasing importance is placed on creating applications that enable efficient management of business processes and reliable data record keeping. This paper presents the development of a web application for managing the services of an autodetailing company, which enables digital tracking of appointments, users, products, and business locations. The application was developed using the Laravel framework, with Tailwind for a responsive user interface, Livewire for interactive components, and Jetstream for secure user authentication.

The system allows user registration independently or through an administrator, management of users divided into different roles, creation and overview of appointments for various types of services, and approval or rejection of appointments by employees and managers. In addition, the application includes management of a web shop with products available in business locations, management of the business locations themselves with their employees and appointment schedules, as well as visualization of business location positions using the Leaflet API. Data is stored in a MySQL relational database, while the user interface is designed to be clear and adapted for everyday use by staff. A REST API has also been implemented to enable programmatic access to user data and integration with external systems.

The application represents a practical solution for digitalizing the operations of an autodetailing company, facilitating the management of appointments, products, and users through a modern, functional, and secure system.

Keywords: web application, autodetailing, appointment management, Laravel, Livewire, Tailwind CSS, Jetstream, REST API, service reservations, user management, Leaflet API, database

Sadržaj

1	Uvod	1
2	Modeliranje aplikacije	2
2.1	Opis aplikacije	2
2.2	Statički UML dijagrami	3
2.2.1	Klasni dijagram	4
2.3	Dinamički UML dijagrami	6
2.3.1	Dijagrami slučajeva korištenja	6
2.3.2	Sekvencijalni dijagrami	8
2.4	ER dijagram baze podataka	13
3	Implementacija	16
3.1	Tehnologija izrade aplikacije	16
3.2	Laravel	16
3.3	MySQL	17
3.4	MVC arhitektura	17
3.5	Objektno-relaciono mapiranje	18
3.6	REST Api	19
4	Analiza rada aplikacije	20
4.1	Opis slučajeva korištenja	20
5	Zaključak	29

Popis slika

1	Klasni dijagram aplikacije za upravljanje autodetailing uslugama	5
2	Dijagram slučajeva korištenja	7
3	Sekvencijalni dijagram za registraciju korisnika	8
4	Sekvencijalni dijagram za prijavu korisnika	9
5	Sekvencijalni dijagram za kreiranje rezervacije od strane klijenta	10
6	Sekvencijalni dijagram za prihvatanje ili odbijanje rezervacije . .	11
7	Sekvencijalni dijagram za upravljanje poslovnim jedinicama . .	12
8	Entity Relationship dijagram	14
9	MVC arhitektura	18
10	Registracija korisnika na sistem	21
11	Neuspješna registracija korisnika	22
12	Prijava korisnika na sistem	23
13	Neuspješna prijava korisnika na sistem	23
14	Kreiranje nove rezervacije	24
15	Neuspješno kreiranje nove rezervacije	25
16	Pregled korisnika unutar sistema	26
17	Uređivanje korisnika od strane administratora	27
18	Pregled proizvoda unutar Web Shop-a	28

1 Uvod

Projekat koji je razvijen predstavlja web aplikaciju za upravljanje uslugama autodetailing firme, s ciljem unaprjeđenja organizacije poslovanja, evidencije korisnika i efikasnog upravljanja rezervacijama termina. U savremenom poslovnom okruženju, autodetailing firme svakodnevno rade sa velikim brojem klijenata, različitim vrstama usluga, više poslovnica i zaposlenika. Tradicionalni načini upravljanja terminima, zasnovani na telefonskim pozivima, papirnim evidencijama ili nepovezanim digitalnim alatima, često dovode do grešaka, preklapanja termina i otežanog praćenja raspoloživosti resursa. Zbog toga se javila potreba za razvojem centralizovanog sistema koji omogućava pouzdano, pregledno i efikasno upravljanje svim aspektima poslovanja autodetailing firme.

Aplikacija je razvijena korištenjem Laravel PHP frameworka, koji omogućava izradu sigurnih, stabilnih i skalabilnih web aplikacija. Laravel koristi MVC (Model–View–Controller) arhitekturu, koja omogućava jasnu podjelu odgovornosti unutar sistema. Modeli omogućavaju rad sa bazom podataka i upravljanje podacima, kontroleri implementiraju poslovnu logiku aplikacije, dok su pogledi odgovorni za prikaz korisničkog interfejsa. Za izradu modernog i responzivnog korisničkog interfejsa korišten je Tailwind CSS, dok je Livewire omogućen za izradu interaktivnih komponenti bez potrebe za pisanjem dodatnog JavaScript koda. Također, Jetstream je korišten za implementaciju autentifikacije korisnika i sigurnosnih mehanizama sistema.

Aplikacija omogućava korisnicima pregled dostupnih usluga i poslovnica, kao i jednostavno kreiranje rezervacija termina. Klijenti mogu rezervisati termine za željene usluge u određenoj poslovnici, dok uposlenici i poslovodje imaju mogućnost pregleda, odobravanja ili odbijanja rezervacija. Administrator ima proširene mogućnosti upravljanja korisnicima, poslovcicama, uslugama i proizvodima koji su dostupni u web shopu. Sistem također omogućava organizaciju poslovnica sa pripadajućim zaposlenicima i terminima, čime se postiže efikasno upravljanje resursima firme.

Za pohranu podataka koristi se MySQL relaciona baza podataka, koja omogućava sigurno i efikasno upravljanje informacija o korisnicima, rezervacijama, uslugama, poslovcicama i proizvodima. Korištenjem Eloquent ORM-a, Laravel omogućava objektno orijentisan pristup radu sa bazom podataka, čime se pojednostavljuje implementacija relacija između entiteta i povećava efikasnost razvoja aplikacije. Također je implementiran REST API koji omogućava programski pristup podacima korisnika i integraciju sa drugim sistemima.

Aplikacija implementira sigurnosne mehanizme kao što su autentifikacija korisnika, autorizacija na osnovu korisničkih uloga i validacija unosa podataka, čime se osigurava da samo ovlašteni korisnici imaju pristup određenim funkcionalnostima sistema. Na ovaj način se povećava sigurnost podataka i pouzdanost rada sistema.

Cilj projekta je razvoj funkcionalne, sigurne i pregledne web aplikacije koja unapređuje organizaciju poslovanja autodetailing firme, pojednostavljuje upravljanje rezervacijama, korisnicima i uslugama, te omogućava efikasno korištenje resursa kroz savremeno i skalabilno softversko rješenje.

2 Modeliranje aplikacije

Modeliranje aplikacije predstavlja jedan od ključnih koraka u procesu razvoja softverskog sistema, jer omogućava jasno definisanje strukture aplikacije, njenih funkcionalnosti i međusobnih odnosa između pojedinih komponenti. Prije same implementacije bilo je neophodno analizirati potrebe autodetailing firme, definisati osnovne funkcionalnosti sistema te predvidjeti na koji način različiti tipovi korisnika komuniciraju s aplikacijom. Ovakav pristup omogućava stabilan razvoj, bolju organizaciju koda i lakše održavanje sistema.

U okviru ovog projekta, modeliranje aplikacije obuhvata izradu UML dijagrama koji prikazuju strukturu sistema, interakciju između korisnika i aplikacije, kao i organizaciju baze podataka. Korišteni su dijagrami slučajeva upotrebe (Use Case), dijagrami klase, sekvensijski dijagrami i ER dijagram, koji zajedno pružaju jasan uvid u funkcionisanje aplikacije. Ovi dijagrami opisuju ključne funkcionalnosti poput autentifikacije korisnika, upravljanja rezervacijama termina, pregleda i odobravanja rezervacija, upravljanja poslovnicama, uslugama i proizvodima, te evidencije aktivnosti korisnika.

Posebna pažnja posvećena je modeliranju relacija između entiteta kao što su korisnici, poslovnice, usluge, rezervacije i proizvodi, kako bi se osigurala konzistentnost i integritet podataka. Također je definisana hijerarhija korisničkih uloga (administrator, poslovodja, uposlenik i klijent), čime su precizno određena prava pristupa i dozvoljene akcije unutar sistema.

Modeliranje sistema omogućava lakše planiranje razvoja aplikacije, smanjuje mogućnost grešaka tokom implementacije i olakšava buduća proširenja funkcionalnosti. Također predstavlja važan dio tehničke dokumentacije koja pomaže u razumijevanju strukture i logike sistema svim učesnicima u razvoju. Na ovaj način postavljeni su čvrsti temelji za izgradnju stabilne, funkcionalne i skalabilne web aplikacije namijenjene upravljanju poslovanjem autodetailing firme, rezervacijama i web shop proizvodima.

2.1 Opis aplikacije

Aplikacija predstavlja web sistem namijenjen upravljanju poslovanjem autodetailing firme, uključujući rezervacije termina, korisnike, poslovnice, usluge i proizvode u web shopu. Cilj aplikacije je omogućiti efikasnu evidenciju i organizaciju poslovnih procesa, praćenje dostupnosti termina, upravljanje zaposlenicima i optimizaciju resursa firme, kao i olakšati administrativne zadatke vezane za upravljanje uslugama i proizvodima. Sistem omogućava centralizovano upravljanje podacima o klijentima, poslovnicama, zaposlenicima, rezervacijama i proizvodima. Svaka rezervacija ili proizvod sadrži detaljne informacije poput naziva, opisa, cijene, kategorije, dostupnosti i pripadajuće poslovnice, čime se osigurava precizna evidencija i bolja organizacija poslovnih aktivnosti.

Aplikacija razlikuje više tipova korisnika, poput administratora, poslovoda, uposlenika i klijenata. Administrator ima proširene privilegije koje uključuju upravljanje korisnicima, poslovnicama, uslugama i proizvodima, kao i pregled svih rezervacija. Poslovodja i uposlenici imaju mogućnost pregleda rezervacija

svoje poslovnice, te odobravanja ili odbijanja termina. Klijenti mogu kreirati i pratiti vlastite rezervacije, pregledavati dostupne usluge i poslovnice, te koristiti web shop za kupovinu proizvoda.

Sistem je osmišljen tako da minimizira mogućnost grešaka u organizaciji termina i evidenciji rezervacija, te da smanji rizik od preklapanja termina ili neadekvatnog korištenja resursa. Automatska validacija unosa podataka, kontrola pristupa bazirana na ulogama i jasna struktura baze podataka doprinose pouzdanosti i sigurnosti sistema. Aplikacija pruža pregledan i responzivan korisnički interfejs, prilagođen radu na različitim uređajima. Korištenjem savremenih tehnologija i principa razvoja softvera, sistem omogućava jednostavno održavanje, proširenje funkcionalnosti i integraciju sa drugim informacionim sistemima u budućnosti.

2.2 Statički UML dijagrami

Statički UML dijagrami predstavljaju način vizualnog prikaza strukture softverskog sistema i njegovih osnovnih elemenata. Ovi dijagrami koriste se za opis arhitekture aplikacije, odnosa između komponenti i organizacije podataka unutar sistema. Njihova glavna svrha je olakšavanje razumijevanja dizajna aplikacije prije same implementacije, kao i pojednostavljinje održavanja i daljeg razvoja sistema. U razvoju softverskih rješenja, UML dijagrami omogućavaju inženjerima da jasno predstave strukturu aplikacije, tokove podataka i međusobne zavisnosti između komponenti. Statički dijagrami fokusirani su na prikaz trajnih elemenata sistema, bez obzira na njihovo ponašanje tokom izvršavanja. U okviru ovog projekta korišteni su sljedeći tipovi statičkih UML dijagrama:

- Dijagram strukture sistema – prikazuje osnovne dijelove aplikacije i njihov međusobni odnos.
- Klasni dijagram – definiše klase, njihove atribute, metode i veze između njih.
- Objektni dijagram – prikazuje konkretnе instance klasa u određenom trenutku.
- Dijagram kompozicije – prikazuje odnose zavisnosti i strukturu složenih komponenti sistema.
- Dijagram ograničenja – opisuje pravila i ograničenja koja važe unutar sistema.

U nastavku rada detaljnije je prikazan i objašnjen klasni dijagram, koji predstavlja osnovu za razumijevanje strukture aplikacije i njenih ključnih funkcionalnosti.

2.2.1 Klasni dijagram

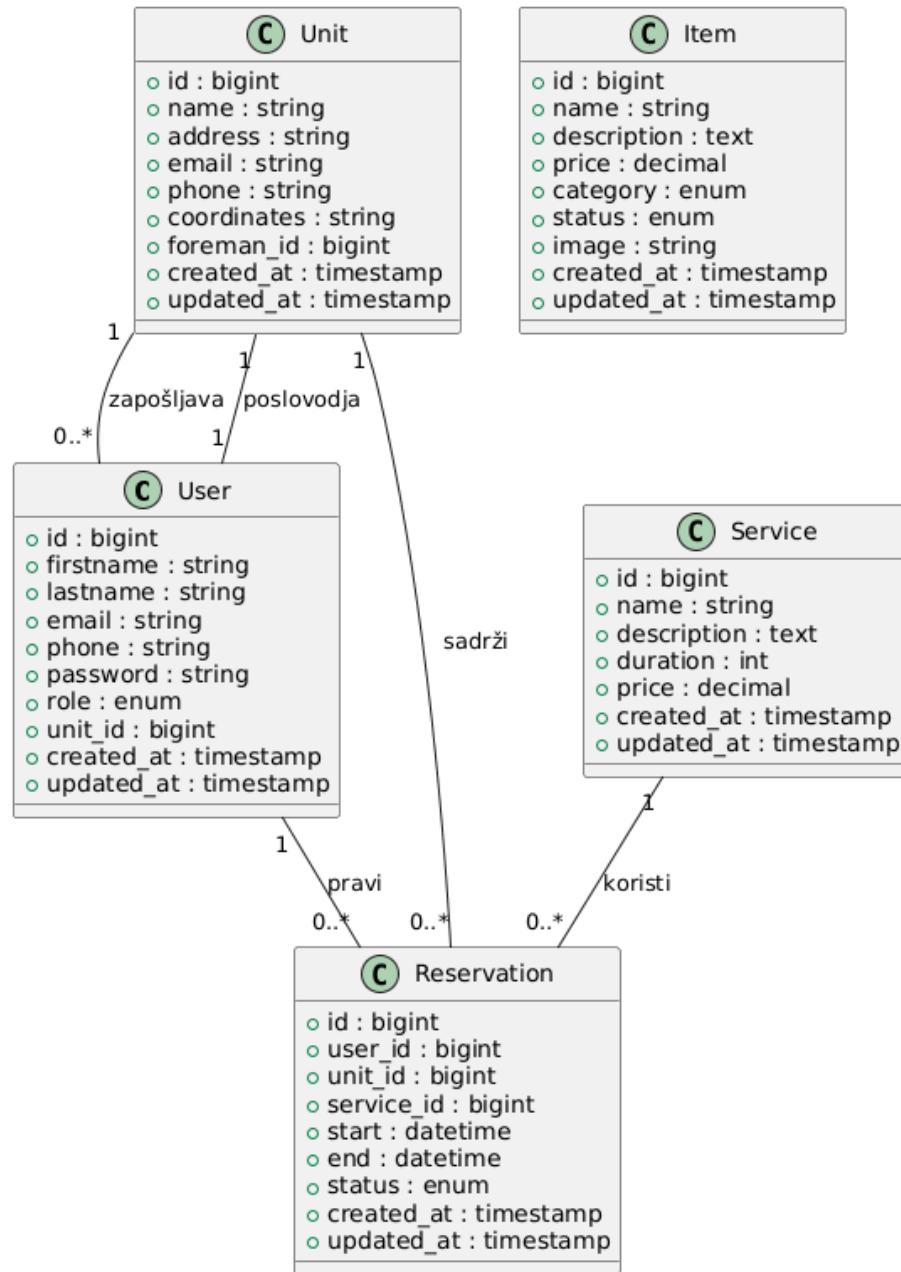
Klasni dijagram prikazuje osnovnu strukturu aplikacije kroz glavne entitete i njihove međusobne odnose. U sistemu se koriste klase *User*, *Unit*, *Service*, *Reservation* i *Item*, koje zajedno omogućavaju upravljanje poslovanjem autodetailing firme, rezervacijama termina, korisnicima, poslovnicama i proizvodima u web shopu.

Klase *Reservation* predstavlja centralni entitet sistema i povezana je sa klasama *User*, *Unit* i *Service*, pri čemu svaka rezervacija pripada jednom korisniku, jednoj poslovničici i jednoj usluzi. Klase *Unit* predstavlja poslovnici i može imati više zaposlenika (*User*) i više rezervacija (*Reservation*), dok je svaki zaposlenik povezan sa jednom poslovnicom. Poslovница može imati definisanog poslovođu (*User*), ali ta veza je opcionalna.

Klase *Item* predstavlja proizvode dostupne u web shopu i može biti povezana sa jednom ili više poslovnica, zavisno od dostupnosti proizvoda. Jedan korisnik (*User*) može kreirati više rezervacija, a sistem omogućava praćenje aktivnosti i statusa rezervacija kroz atribute klase *Reservation*.

Ovakva struktura omogućava efikasno upravljanje rezervacijama, korisnicima, poslovnicama i proizvodima, te pruža fleksibilan i skalabilan temelj za dalji razvoj i proširenje funkcionalnosti aplikacije. Klasni dijagram jasno prikazuje hijerarhiju korisničkih uloga (administrator, poslovođa, uposlenik, klijent) i relacije između entiteta, čime se olakšava održavanje i nadogradnja sistema.

Klasni dijagram aplikacije prikazan je na slici 1.



Slika 1: Klasni dijagram aplikacije za upravljanje autodetailing uslugama

2.3 Dinamički UML dijagrami

Dinamički UML dijagrami opisuju ponašanje sistema tokom vremena i prikazuju način na koji se njegove komponente međusobno povezuju i reaguju na određene događaje. Ovi dijagrami omogućavaju razumijevanje toka izvršavanja procesa, razmjene podataka i interakcije između korisnika i sistema. Posebno su značajni za prikaz realnog ponašanja aplikacije tokom izvršavanja pojedinih funkcionalnosti. Dinamički dijagrami se koriste za modeliranje vremenskih sekvenci, promjena stanja sistema, kao i komunikacije između različitih komponenti. Na taj način omogućavaju detaljniji uvid u rad aplikacije i pomažu u identifikaciji potencijalnih problema tokom implementacije. U okviru UML standarda razlikuje se nekoliko vrsta dinamičkih dijagrama:

- Dijagram aktivnosti,
- Dijagram stanja,
- Dijagram slučajeva korištenja,
- Sekvencijalni dijagram,
- Komunikacioni dijagram.

U nastavku rada detaljnije su opisani i prikazani dijagram slučajeva korištenja i sekvencijalni dijagram, jer oni najjasnije predstavljaju interakciju korisnika sa sistemom i tok izvršavanja osnovnih funkcionalnosti aplikacije.

2.3.1 Dijagrami slučajeva korištenja

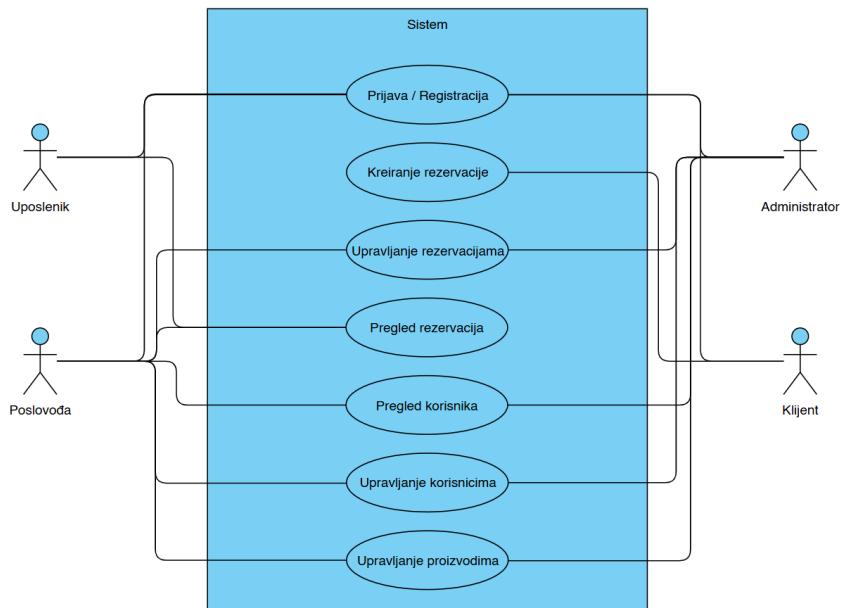
Dijagrami slučajeva korištenja (engl. *Use Case Diagram*) koriste se za prikaz funkcionalnosti sistema iz perspektive krajnjih korisnika i njihove interakcije sa aplikacijom. U okviru ovog projekta, ovi dijagrami omogućavaju jasan pregleđ načina na koji različiti tipovi korisnika koriste sistem i koje radnje mogu izvršavati unutar aplikacije.

Use case dijagrami se koriste za ilustraciju osnovnih funkcionalnih zahtjeva sistema, bez ulaska u tehničke detalje implementacije. Njihova svrha je da prikažu scenarije korištenja sistema, uloge korisnika i granice sistema, čime se olakšava razumijevanje ponašanja aplikacije u realnim uslovima.

U aplikaciji *AutoDetailManager* definisana su četiri tipa aktera: *Klijent*, *Uposlenik*, *Poslovođa* i *Administrator*.

Klijent ima mogućnost registracije ili prijave na sistem, pregledavanja dostupnih usluga i poslovnica, kreiranja i praćenja vlastitih rezervacija, te korištenja web shopa za kupovinu proizvoda. Uposlenik, pored navedenih funkcionalnosti, može pregledavati i upravljati rezervacijama unutar svoje poslovnice, odobravati ili odbijati termine i unositi napomene o izvršenim uslugama. Poslovođa ima širi pristup, uključujući pregled i upravljanje svim rezervacijama u svojoj poslovnici, praćenje rada uposlenika i odobravanje termina. Administrator ima najviši nivo privilegija i, pored svih prethodnih mogućnosti, može upravljati korisnicima sistema, dodavati ili uređivati poslovnice, usluge i proizvode, te upravljati cjelokupnim sadržajem web shopa i evidencijom aktivnosti korisnika.

Prikazani dijagram slučajeva korištenja razvijene aplikacije prikazan je na slici 2.



Slika 2: Dijagram slučajeva korištenja

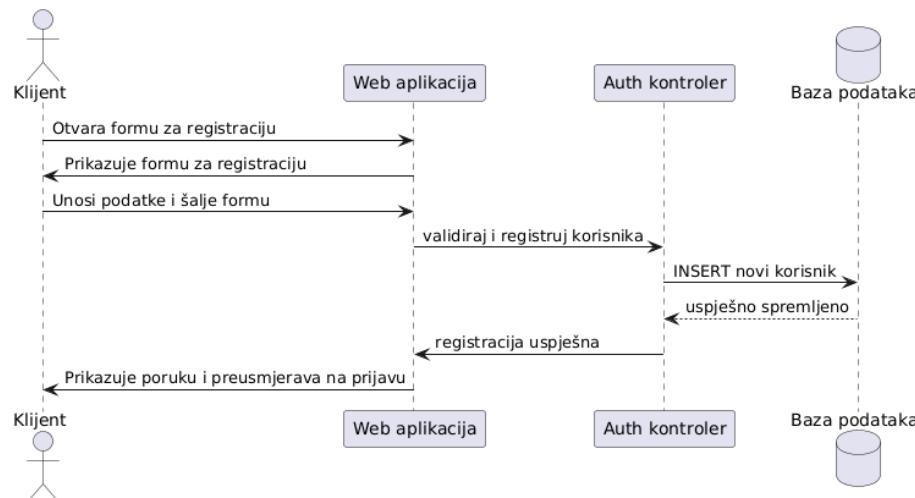
Dijagram slučajeva korištenja pruža pregled ključnih funkcionalnosti aplikacije i predstavlja osnovu za dalju analizu sistema, izradu ostalih UML dijagrama i implementaciju poslovne logike aplikacije.

2.3.2 Sekvencijalni dijagrami

Sekvencijalni dijagram koristi se za prikaz vremenskog slijeda interakcija između aktera i sistema, odnosno načina na koji se razmjenjuju poruke tokom izvršavanja određene funkcionalnosti. Ovaj dijagram omogućava jasno razumijevanje redoslijeda aktivnosti i komunikacije između korisnika, kontrolera, modela i baze podataka, bez ulaska u tehničke detalje implementacije.

Na slici 3 prikazan je sekvencijalni dijagram za registraciju korisnika. Klijent unosi svoje podatke za registraciju putem forme, uključujući ime, prezime, email, telefon i lozinku. Nakon slanja forme, zahtjev se proslijeđuje *AuthController*-u. Kontroler vrši validaciju podataka i provjerava da li email već postoji u bazi putem modela *User*.

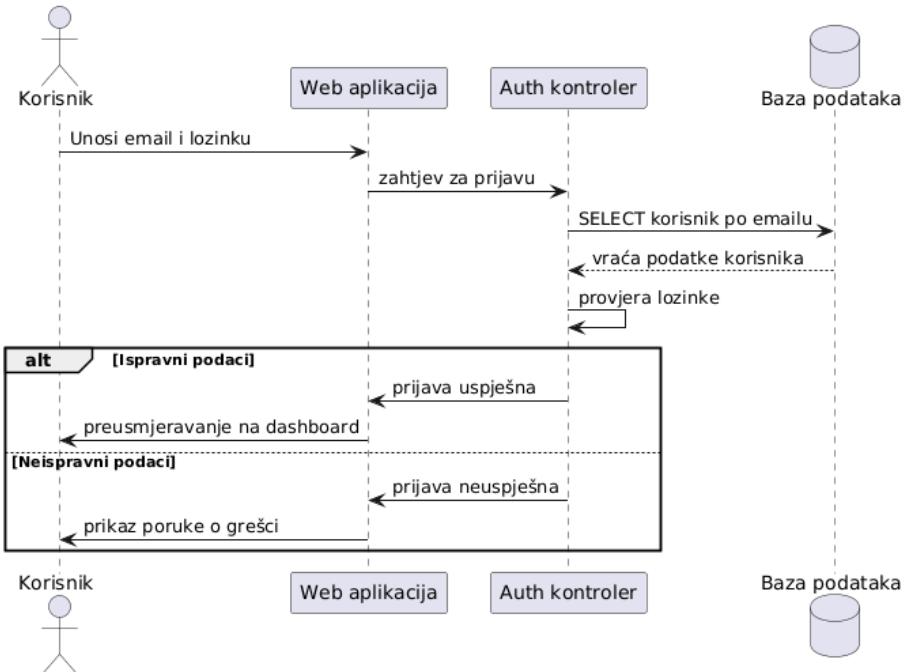
U slučaju da su podaci neispravni ili email već postoji, sistem vraća poruku o grešci i korisnik je obaviješten da ispravi unos. U slučaju da su svi podaci ispravni, kreira se novi korisnički nalog u bazi podataka, a sistem automatski kreira sesiju za korisnika i preusmjerava ga na početnu stranicu aplikacije ili stranicu sa potvrdom registracije.



Slika 3: Sekvencijalni dijagram za registraciju korisnika

Na slici 4 prikazan je sekvencijalni dijagram za prijavu korisnika. Klijent unosi svoje pristupne podatke putem login forme, uključujući email i lozinku. Nakon slanja forme, zahtjev se proslijeđuje *AuthController*-u. Kontroler provjerava unose i upoređuje podatke sa zapisima u bazi putem modela *User*.

Ukoliko su podaci neispravni, sistem vraća poruku o grešci i korisnik je obaviješten da ponovo unese podatke. U slučaju uspješne autentifikacije, kreira se sesija za korisnika i on se preusmjerava na početnu stranicu aplikacije.

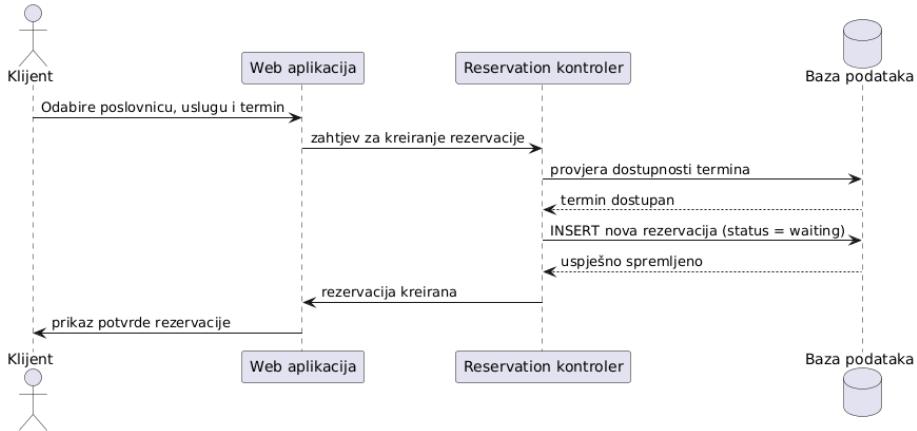


Slika 4: Sekvencijalni dijagram za prijavu korisnika

Na slici 9 prikazan je sekvencijalni dijagram za kreiranje rezervacije termina. Klijent odabirom željene usluge i poslovnice putem forme šalje zahtjev *ReservationController*-u. Kontroler provjerava dostupnost termina u bazi podataka putem modela *Reservation* i modela *Unit*.

Ukoliko termin nije dostupan, sistem vraća poruku o grešci i klijent je obaviješten da odabere drugi termin. Ako je termin slobodan, rezervacija se pohranjuje u bazu podataka. Nakon uspješnog kreiranja rezervacije, klijentu se prikazuje potvrda i status rezervacije, a poslovoda odgovoran za poslovnicu može pregledati i odobriti ili odbiti rezervaciju.

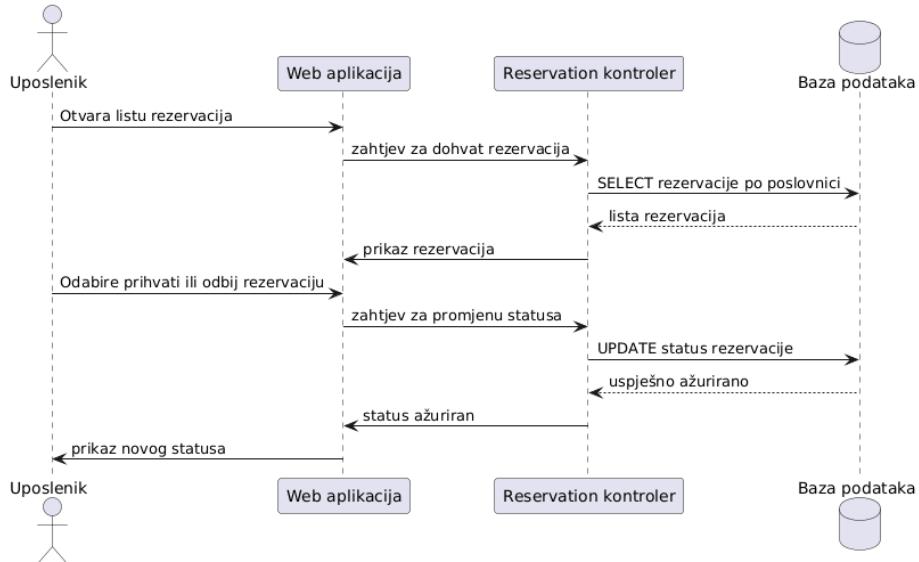
Ovaj sekvencijalni dijagram jasno prikazuje redoslijed interakcija između klijenta, kontrolera, modela i baze podataka, olakšavajući razumijevanje procesa kreiranja rezervacija i osiguravajući pravilno upravljanje terminima u aplikaciji.



Slika 5: Sekvencijalni dijagram za kreiranje rezervacije od strane klijenta

Na slici 6 prikazan je sekvencijalni dijagram za odobravanje ili odbijanje rezervacije termina od strane Poslovode. Nakon što klijent kreira rezervaciju, zahtjev za pregled rezervacija dolazi do *ReservationController-a*, koji dohvata relevantne podatke o rezervaciji iz baze podataka putem modela *Reservation*. Poslovoda pregledava detalje rezervacije i donosi odluku da li će rezervaciju odobriti ili odbiti. Odluka se šalje nazad *ReservationController-u*, koji ažurira status rezervacije u bazi podataka.

Nakon izvršenih promjena, sistem obavještava klijenta o statusu rezervacije putem prikaza potvrde ili poruke o odbijanju. Sekvencijalni dijagram jasno prikazuje redoslijed interakcija između klijenta, uposlenika, poslovode, kontrolera i baze podataka, omogućavajući pregled toka procesa od kreiranja rezervacije do njenog konačnog odobrenja ili odbijanja.

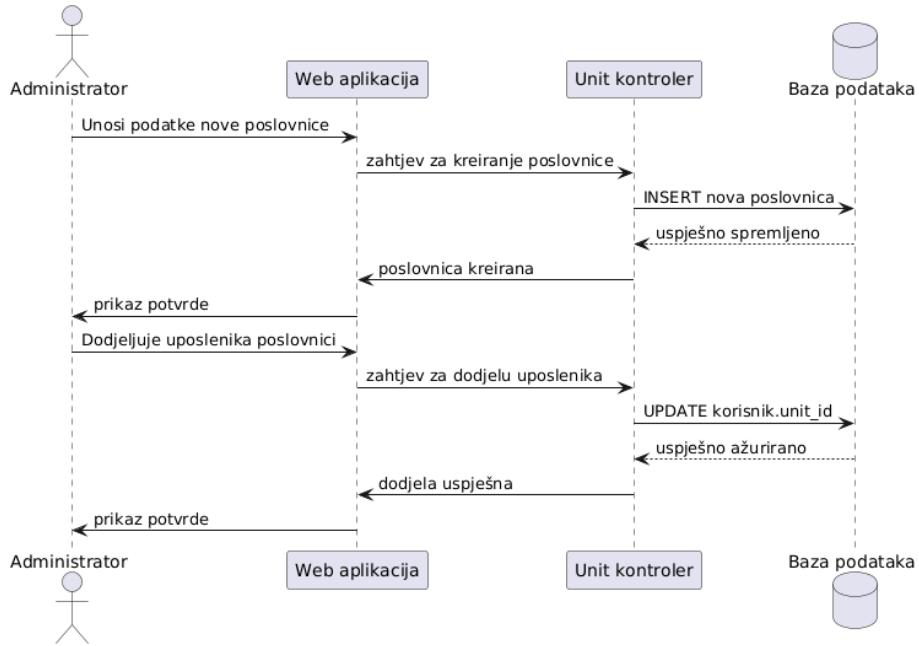


Slika 6: Sekvencijalni dijagram za prihvatanje ili odbijanje rezervacije

Na slici 7 prikazan je sekvencijalni dijagram za upravljanje poslovnim jedinicama u aplikaciji. Administrator kreira, uređuje ili briše poslovnici putem odgovarajuće forme. Nakon slanja zahtjeva, *UnitController* obrađuje podatke i provjerava njihovu validnost.

U slučaju neispravnog unosa, sistem vraća poruku o grešci i administrator je obaviješten da ispravi podatke. Ako su podaci ispravni, promjene se pohranjuju u bazu podataka putem modela *Unit*. Nakon uspješnog izvršenja operacije, sistem prikazuje poruku o uspjehu i osvježava prikaz liste poslovnica kako bi administrator mogao vidjeti ažurirane podatke.

Sekvencijalni dijagram jasno prikazuje redoslijed interakcija između administratora, kontrolera i baze podataka, olakšavajući razumijevanje procesa upravljanja poslovnim jedinicama i omogućavajući pravilnu implementaciju funkcionalnosti unutar aplikacije.



Slika 7: Sekvencijalni dijagram za upravljanje poslovnim jedinicama

Prikazani sekvencijalni dijagrami ilustruju način na koji sistem obrađuje korisničke zahtjeve, redoslijed izvršavanja operacija i komunikaciju između slojeva aplikacije. Ovi dijagrami doprinose boljem razumijevanju dinamike sistema i predstavljaju važan dio tehničke dokumentacije aplikacije.

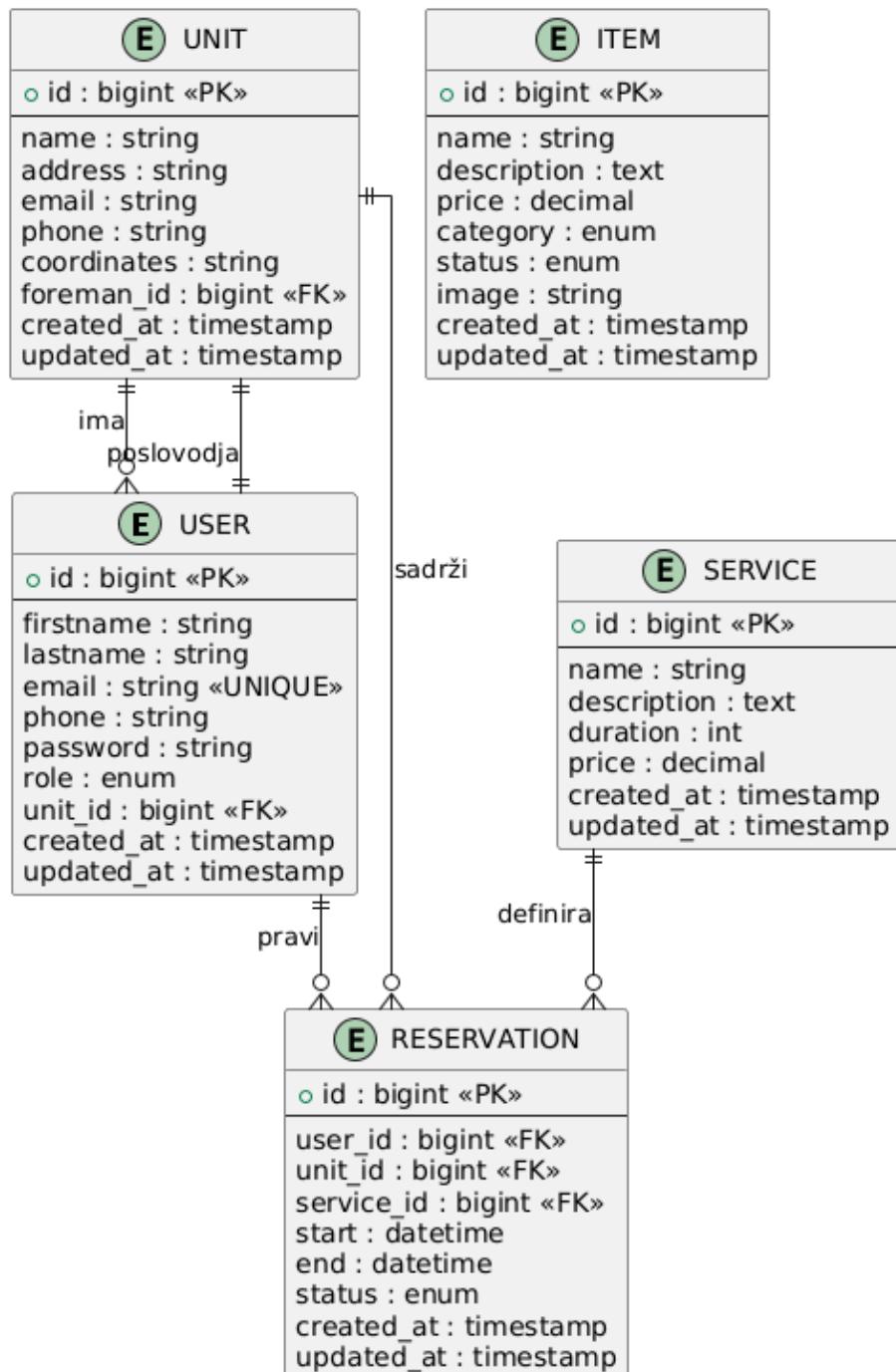
2.4 ER dijagram baze podataka

Baza podataka aplikacije za upravljanje autodetailing uslugama opisana je pomoću ER (Entity Relationship) dijagrama, koji prikazuje osnovne entitete i njihove međusobne veze. Sistem se sastoji od ključnih tabela: *users*, *units*, *services*, *reservations* i *items*. Centralni entitet u bazi je tabela *reservations*, koja sadrži informacije o rezervacijama termina, uključujući datum i vrijeme početka i završetka, status rezervacije, te povezane podatke o korisniku, poslovnici i usluzi.

Entitet *users* služi za evidenciju svih korisnika sistema, uključujući klijente, uposlenike, poslovođe i administratore, sa podacima kao što su ime, prezime, email, telefon, lozinka i uloga. Tabela *units* predstavlja poslovnice autodetailing firme i sadrži informacije o nazivu, adresi, emailu, telefonu i lokaciji. Svaka poslovница može imati više zaposlenih i više rezervacija.

Entitet *services* definiše sve dostupne vrste usluga koje se nude u poslovnica, uključujući naziv, opis, cijenu i trajanje usluge. Entitet *items* predstavlja proizvode dostupne u web shopu, sa podacima o nazivu, opisu, cijeni, kategoriji, statusu i slici proizvoda.

Tabela *reservations* povezuje korisnike, poslovnice i usluge, omogućavajući praćenje termina i statusa odobravanja ili odbijanja rezervacija. Ovakva struktura baze omogućava precizno upravljanje korisnicima, poslovcima, terminima i proizvodima, te olakšava implementaciju funkcionalnosti za rezervacije, upravljanje uslugama i web shop unutar aplikacije.



Slika 8: Entity Relationship dijagram

Veze između entiteta u aplikaciji su uglavnom tipa 1:N. Jedna poslovница (*unit*) može imati više zaposlenih (*users*) i više rezervacija (*reservations*), dok svaka rezervacija pripada tačno jednoj poslovničici i jednom korisniku. Svaka rezervacija je vezana i za tačno jednu uslugu (*service*), dok jedna usluga može biti povezana sa više rezervacija.

Svi proizvodi (*items*) u web shopu su samostalni entiteti, pri čemu jedan proizvod može biti dostupan u više poslovnica, a svaka poslovница može imati više proizvoda. Jedan korisnik može kreirati više rezervacija, ali svaka rezervacija pripada samo jednom korisniku.

Ovakva struktura omogućava jednostavno upravljanje korisnicima, poslovnicama, uslugama i rezervacijama, jasno definisane odnose između osnovnih elemenata sistema i pouzdano praćenje termina i dostupnosti proizvoda unutar aplikacije.

3 Implementacija

3.1 Tehnologija izrade aplikacije

Aplikacija je razvijena korištenjem programskog jezika PHP uz primjenu Laravel frameworka, koji predstavlja jedan od najpopularnijih alata za razvoj modernih web aplikacija. Laravel je zasnovan na MVC (engl. Model–View–Controller) arhitekturi, koja omogućava jasno razdvajanje poslovne logike, korisničkog interfejsa i upravljanja podacima, čime se postiže bolja organizacija koda i lakše održavanje sistema.

Korištenjem ovog arhitektonskog obrasca omogućena je veća preglednost aplikacije, jednostavnije testiranje i lakša nadogradnja funkcionalnosti. Modeli su zaduženi za rad s podacima i komunikaciju s bazom, kontroleri obrađuju korisničke zahtjeve, dok su prikazi odgovorni za vizuelnu prezentaciju podataka korisniku.

Za razvoj korisničkog interfejsa korišten je Tailwind CSS, koji omogućava responzivan i moderan dizajn, dok su interaktivne komponente implementirane korištenjem Livewire-a. Sigurnost autentifikacije i upravljanje sesijama korisnika ostvareno je korištenjem Jetstream paketa. Za prikaz lokacija poslovnica i interakciju sa mapama korišten je Leaflet API.

Za upravljanje bazom podataka korišten je MySQL sistem za upravljanje bazama podataka (DBMS), koji omogućava pouzdano, efikasno i sigurno čuvanje podataka. MySQL pruža stabilno okruženje za rad sa relacionim bazama, što je posebno pogodno za aplikacije koje upravljaju većim brojem povezanih entiteta, kao što su korisnici, poslovnice, rezervacije, usluge i proizvodi web shopa.

Kombinacijom Laravel frameworka, MySQL baze podataka i modernih front-end tehnologija osigurana je stabilna, skalabilna i efikasna osnova za razvoj ovakve web aplikacije, koja omogućava upravljanje korisnicima, poslovnicama, terminima i proizvodima na intuitivan i siguran način.

3.2 Laravel

Laravel je open-source PHP framework namijenjen razvoju modernih i skalabilnih web aplikacija. Odlikuje se jednostavnom sintaksom, jasno definisanim struktrom i bogatim skupom ugrađenih funkcionalnosti koje značajno ubrzavaju proces razvoja. Jedna od njegovih ključnih prednosti je implementacija objektnog relacionog mapiranja (ORM) putem Eloquent ORM-a, koji omogućava intuitivnu i sigurnu komunikaciju sa bazom podataka.

Laravel pruža napredne mehanizme za upravljanje rutama, autentifikacijom, autorizacijom, validacijom podataka i sigurnošću aplikacije. Također nudi podršku za izradu modularnih komponenti i korištenje Blade templating sistema, koji omogućava jednostavno kreiranje dinamičkih korisničkih interfejsa.

Zahvaljujući svojoj fleksibilnosti i bogatoj dokumentaciji, Laravel predstavlja pouzdan izbor za razvoj savremenih web aplikacija, omogućavajući bržu implementaciju funkcionalnosti, lakše održavanje koda i visok nivo sigurnosti aplikacije.

3.3 MySQL

MySQL je relacijski sistem za upravljanje bazama podataka koji omogućava efikasno čuvanje, obradu i organizaciju podataka. Podaci su strukturirani u tabele sastavljene od redova i kolona, pri čemu svaka tabela predstavlja određeni entitet unutar sistema.

Zahvaljujući svojoj pouzdanosti, brzini i širokoj primjeni, MySQL je jedan od najčešće korištenih sistema za upravljanje bazama podataka u web aplikacijama.

U okviru ovog projekta, MySQL se koristi za pohranu podataka o korisnicima, poslovnicama, uslugama, rezervacijama i proizvodima web shopa, omogućavajući sigurno i efikasno upravljanje informacijama i olakšavajući implementaciju funkcionalnosti vezanih za termine, usluge i prodaju proizvoda.

3.4 MVC arhitektura

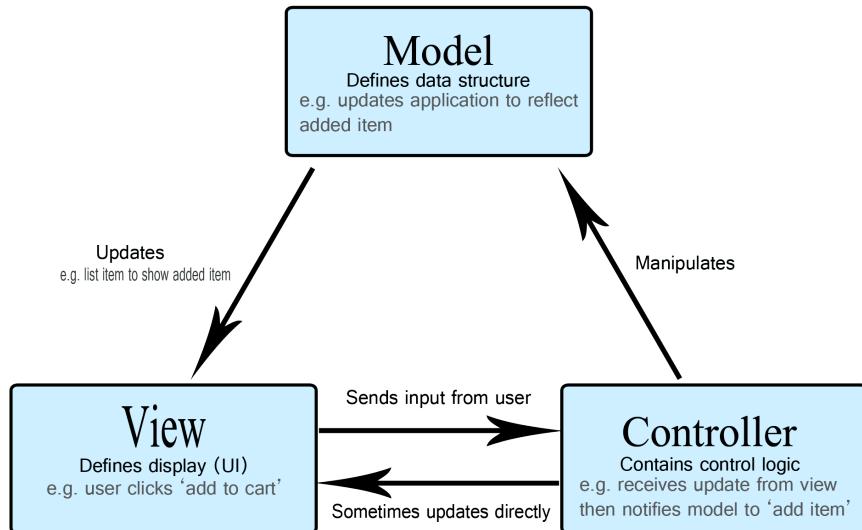
Model–View–Controller (MVC) predstavlja arhitektonski obrazac koji se koristi u razvoju softverskih aplikacija s ciljem jasnog razdvajanja odgovornosti unutar sistema. Ovaj pristup omogućava bolju organizaciju koda, lakše održavanje i jednostavnije proširivanje funkcionalnosti aplikacije.

Model predstavlja sloj zadužen za obradu podataka i poslovnu logiku aplikacije. On upravlja komunikacijom sa bazom podataka, definiše pravila rada sistema i omogućava pristup podacima kroz jasno definisane metode. U kontekstu ovakve aplikacije, modeli obuhvataju entitete kao što su korisnici (*users*), poslovnice (*units*), usluge (*services*), rezervacije (*reservations*) i proizvodi (*items*).

View (pogled) je zadužen za prikaz podataka korisniku i predstavlja korisnički interfejs aplikacije. Ovaj sloj ne sadrži poslovnu logiku, već isključivo prikazuje informacije dobijene od modela, koristeći različite vizuelne komponente poput formi, tabela, listi i interaktivnih komponenti kreiranih pomoću Livewire-a. Jedan model može imati više različitih pogleda, što omogućava fleksibilan i prilagodljiv prikaz sadržaja.

Controller (kontroler) predstavlja vezu između modela i pogleda. Njegova uloga je da obrađuje korisničke zahtjeve, poziva odgovarajuće metode modela i određuje koji će se pogled prikazati korisniku. Kontroler time upravlja tokom aplikacije i omogućava pravilnu komunikaciju između svih komponenti sistema.

Primjena MVC arhitekture donosi brojne prednosti, uključujući lakše testiranje, bolju organizaciju koda i jednostavnije održavanje aplikacije. Zbog svoje fleksibilnosti i skalabilnosti, MVC obrazac je široko prihvaćen u modernim web okvirima poput Laravel frameworka, koji je korišten u razvoju ove aplikacije, čime se osigurava pouzdana i skalabilna osnova za upravljanje korisnicima, poslovnicama, terminima i proizvodima.



Slika 9: MVC arhitektura

Kao što se može vidjeti na slici 5, prikazana je MVC arhitektura te komunikacija između komponenti, tačnije modela, pogleda i kontrolera.

3.5 Objektno-relaciono mapiranje

Objektno-relaciono mapiranje (ORM – Object Relational Mapping) predstavlja mehanizam koji omogućava rad s bazom podataka kroz objekte i klase unutar aplikacije, bez potrebe za direktnim pisanjem SQL upita. ORM omogućava povezivanje objekata iz programske jezike sa tabelama u bazi podataka, čime se pojednostavljuje upravljanje podacima i unapređuje čitljivost i održivost koda.

Korištenjem ORM-a, programeri mogu izvršavati operacije poput dodavanja, ažuriranja, brisanja i čitanja podataka pomoću metoda i objekata, dok se generisanje SQL upita odvija automatski u pozadini. Na taj način olakšava se rad s bazom podataka i smanjuje mogućnost grešaka prilikom ručnog pisanja upita. ORM takođe omogućava lakše testiranje poslovne logike, jer aplikacija nije direktno vezana za konkretnu implementaciju baze podataka.

Dodatna prednost ORM pristupa jeste mogućnost jednostavne promjene sistema za upravljanje bazom podataka, na primjer prelazak sa MySQL-a na PostgreSQL, bez potrebe za značajnim izmjenama u aplikacijskom kodu. ORM djeluje kao posrednik između aplikacije i baze.

Programeri na taj način mogu raditi s podacima na apstraktnom nivou, bez direktnog upravljanja složenim SQL strukturama. U okviru ovog projekta korišten je ORM koji je sastavni dio Laravel frameworka – Eloquent ORM. Eloquent omogućava jednostavno mapiranje tabele baze podataka u modele, definisanje relacija između entiteta i efikasno upravljanje podacima, čime značajno doprinosi brzini razvoja, čitljivosti koda i stabilnosti aplikacije.

3.6 REST Api

REST API (Representational State Transfer – Application Programming Interface) predstavlja arhitektonski stil za izgradnju web servisa koji omogućava komunikaciju između klijenta i servera korištenjem standardnih HTTP metoda. Ovaj pristup zasniva se na principima jednostavnosti, skalabilnosti i bezstanj-skog (stateless) rada, pri čemu svaki resurs unutar sistema, poput korisnika, poslovnica, usluga, rezervacija ili proizvoda, ima jedinstveni identifikator (URI – Uniform Resource Identifier).

Klijent komunicira sa serverom koristeći standardne HTTP metode kao što su GET, POST, PUT i DELETE, koje omogućavaju dohvati, kreiranje, ažuriranje i brisanje podataka. Razmjena podataka se najčešće vrši u JSON formatu, što omogućava jednostavnu obradu, čitljivost i kompatibilnost sa različitim platfor-mama i tehnologijama.

REST API omogućava modularan i fleksibilan dizajn sistema, gdje su funkcionalnosti jasno razdvojene i lako proširive. Ovakav pristup olakšava održavanje aplikacije, poboljšava njenu skalabilnost i omogućava jednostavnu integraciju sa drugim servisima ili klijentskim aplikacijama, uključujući web i mobilne plat-forme.

Zbog svoje jednostavnosti, pouzdanosti i široke primjene, REST API pred-stavlja standardno rješenje u razvoju savremenih web aplikacija, uključujući i ovu aplikaciju, gdje omogućava efikasnu komunikaciju između korisničkog interfejsa i serverske logike, posebno za upravljanje korisnicima, poslovnicama, terminima, uslugama i proizvodima web shopa.

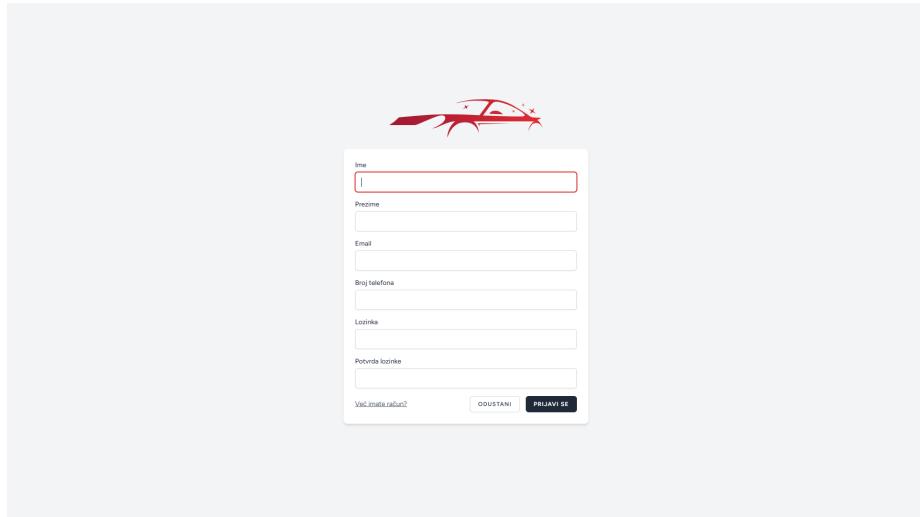
4 Analiza rada aplikacije

Analiza rada web aplikacije izvršena je kroz definisanje ključnih slučajeva korištenja sistema. Za svaki slučaj identificirani su akteri, definisani preduslovi, osnovni tok izvršavanja te eventualni sporedni scenariji u slučaju greške. Ova analiza omogućava bolje razumijevanje funkcionalnosti aplikacije, olakšava testiranje i planiranje daljeg razvoja sistema.

4.1 Opis slučajeva korištenja

Slučaj korištenja 1: Registracija korisnika

- Naziv SK: Registracija korisnika,
- Akteri SK: Novi korisnik, Administrator (ako dodaje korisnika),
- Učesnici SK: Korisnik i sistem,
- Preduslov: Korisnik nema prethodno kreiran nalog ili Administrator dodaje korisnika,
- Osnovni scenarij SK:
 - Korisnik popunjava registracijsku formu unosom imena, prezimena, e-mail adrese, lozinke i ostalih traženih podataka,
 - Korisnik klikne na dugme *Register* ili Administrator potvrdi dodavanje korisnika,
 - Sistem provjerava ispravnost i validnost unesenih podataka (npr. jedinstvenost e-maila, snagu lozinke),
 - U slučaju greške, sistem prikazuje odgovarajuće poruke i traži ispravku,
 - Ako su podaci validni, sistem kreira novi korisnički račun i čuva podatke u bazi,
 - Sistem automatski autentificira korisnika (ako se registracija vrši samostalno) i preusmjerava ga na Dashboard ili potvrđuje uspješnu registraciju.



Slika 10: Registracija korisnika na sistem

- Sporedni scenarij SK:

- Korisnik unese neispravne ili nepotpune podatke (npr. prazna obavezna polja),
- Sistem vrši validaciju i otkriva greške u unosu,
- Sistem odbija registraciju korisnika,
- Prikazuje se poruka o grešci (npr. e-mail već postoji, neispravan format e-maila ili lozinka ne zadovoljava sigurnosne uslove),
- Korisnik ispravlja podatke i ponovo pokušava registraciju.

The screenshot shows a registration form for a system. At the top, there is a logo of a red car with a star above it. Below the logo, the text "Desila se greška." (An error occurred) is displayed in red. Underneath, a bullet point says "Lozinka se ne podudara." (The password does not match). The form fields are as follows:

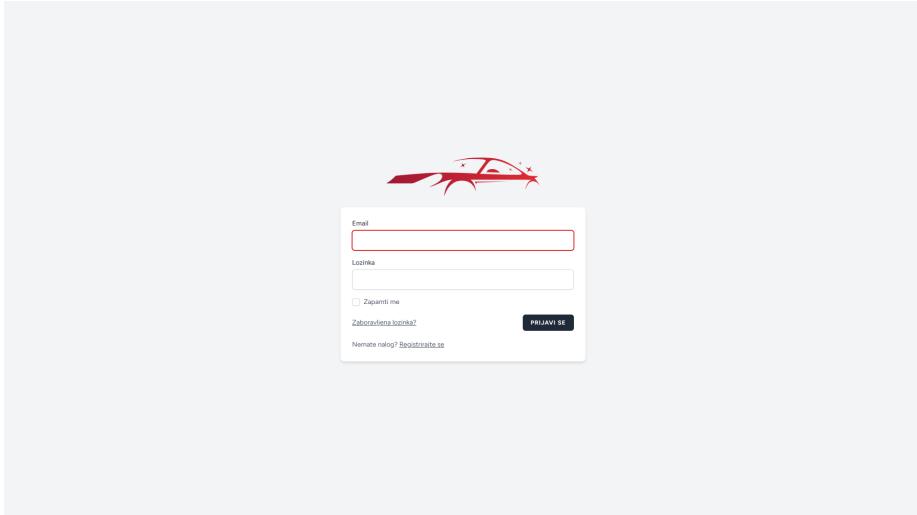
Ime	Hodis
Prezime	Mahmutović
Email	hadis@example.com
Broj telefona	062345678
Lozinka	(empty)
Potvrda lozinke	(empty)

At the bottom of the form, there are two buttons: "ODUSTANI" (Cancel) and "PRIJAVA SE" (Log In).

Slika 11: Neuspješna registracija korisnika

Slučaj korištenja 2: Prijava na sistem

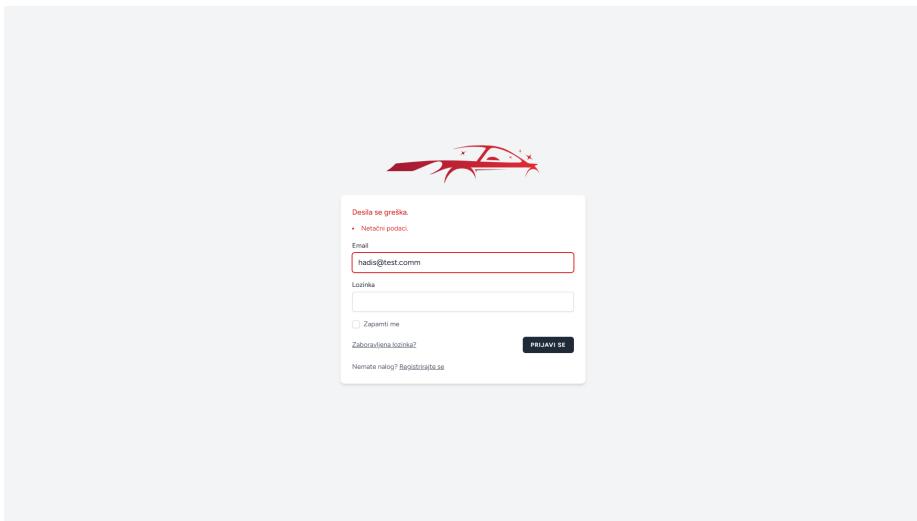
- Naziv SK: Prijava na sistem,
- Akteri SK: Klijent, Uposlenik, Poslovođa, Administrator,
- Učesnici SK: Korisnik i sistem,
- Preduslov: Korisnik posjeduje registrovan korisnički račun,
- Osnovni scenarij SK:
 - Korisnik pristupa stranici za prijavu,
 - Korisnik unosi e-mail adresu i lozinku,
 - Korisnik potvrđuje prijavu klikom na dugme *Prijava*,
 - Sistem provjerava ispravnost unesenih kredencijala,
 - Sistem autentificira korisnika,
 - Sistem preusmjerava korisnika na početnu stranicu sistema u skladu sa njegovom ulogom.



Slika 12: Prijava korisnika na sistem

- Sporedni scenarij SK:

- Korisnik uneće netočne podatke,
- Sistem odbije prijavu,
- Sistem prikazuje poruku o grešci,
- Korisnik ponovo pokušava prijavu.



Slika 13: Neuspješna prijava korisnika na sistem

Slučaj korištenja 3: Kreiranje rezervacije

- Naziv SK: Kreiranje rezervacije,
- Akteri SK: Klijent,
- Učesnici SK: Klijent i sistem,
- Preduslov: Klijent je prijavljen na sistem,
- Osnovni scenarij SK:
 - Klijent pristupa formi za kreiranje rezervacije,
 - Klijent bira željeni proizvod ili uslugu,
 - Klijent unosi potrebne podatke (datum, vrijeme, dodatne napomene),
 - Klijent potvrđuje kreiranje rezervacije,
 - Sistem validira unesene podatke,
 - Sistem kreira rezervaciju i sprema je u bazu podataka,
 - Sistem prikazuje potvrdu o uspješnom kreiranju rezervacije.

Slika 14: Kreiranje nove rezervacije

- Sporedni scenarij SK:
 - Uneseni podaci nisu validni ili termin nije dostupan,
 - Sistem odbija kreiranje rezervacije,

- Sistem prikazuje poruku o grešci.

The screenshot shows a modal window titled "Kreiranje nove rezervacije" (Create new reservation). The window has a header "Popunite podatke za rezervaciju termina". It includes fields for "Poslovna jedinica" (Business unit) set to "Paucek LLC", a map showing a location with a blue marker, and a "Usluga" (Service) dropdown menu with the message "...Odaberite uslugu -- Please select an item in the list.". Below the service dropdown is a "Početak rezervacije" (Start of reservation) date and time set to "02/27/2026 10:45 AM". At the bottom right of the modal is a green button labeled "KREIRAJ REZERVACIJU" (Create reservation).

Slika 15: Neuspješno kreiranje nove rezervacije

Slučaj korištenja 4: Pregled rezervacija

- Naziv SK: Pregled rezervacija,
- Akteri SK: Klijent, Uposlenik, Poslovođa, Administrator,
- Učesnici SK: Korisnik i sistem,
- Preduslov: Korisnik je prijavljen na sistem,
- Osnovni scenarij SK:
 - Korisnik pristupa stranici za pregled rezervacija,
 - Sistem dohvaća rezervacije iz baze podataka,
 - Sistem prikazuje listu rezervacija,
 - Korisnik može pregledati detalje pojedinačne rezervacije.

Slučaj korištenja 5: Upravljanje rezervacijama

- Naziv SK: Upravljanje rezervacijama,
- Akteri SK: Poslovođa, Administrator,
- Učesnici SK: Korisnik i sistem,

- Preduslov: Korisnik je prijavljen i ima odgovarajuće privilegije,
- Osnovni scenarij SK:
 - Korisnik pristupa sekciji za upravljanje rezervacijama,
 - Sistem prikazuje listu rezervacija,
 - Korisnik bira rezervaciju za uređivanje ili brisanje,
 - Korisnik vrši izmjene ili potvrđuje brisanje,
 - Sistem ažurira ili briše rezervaciju u bazi,
 - Sistem prikazuje potvrdu o uspješnoj operaciji.

Slučaj korištenja 6: Pregled korisnika

- Naziv SK: Pregled korisnika,
- Akteri SK: Poslovođa, Administrator,
- Učesnici SK: Korisnik i sistem,
- Preduslov: Korisnik ima administratorske ili poslovodne privilegije,
- Osnovni scenarij SK:
 - Korisnik pristupa sekciji za pregled korisnika,
 - Sistem dohvaća listu korisnika iz baze,
 - Sistem prikazuje listu korisnika sa njihovim osnovnim podacima i ulogama.

The screenshot shows a user interface for managing users. At the top, there is a navigation bar with links: Kalendar, Web Shop, Korisnici (which is underlined), and Poslovne jedinice. On the right side of the header, it says 'Hadi Mahmutović' with a dropdown arrow. Below the header, there is a search bar labeled 'Pretrazi korisnike...' and a button '+ DODAJ KORISNIKA'. The main content area is titled 'Evidencija korisnika' and contains a table with the following data:

IME I	PRIJEME	EMAIL	TELEFON	OVLAST	POSLOVNA JEDINICA	AKCJE
Dawn	Zulaf	ubernier@example.net	+1-909-685-3199	Klijent	-	
Demetrius	Rath	naomie.johnson@example.net	352-798-7858	Poslovnik	Gulgowski Group	
Geirtrude	Connely	bpfeffer@example.net	682-6712045	Poslovnik	Gulgowski Group	
Hadi	Mahmutović	hadi@test.com	0666779627	Administrator	-	
Jenel	Stoltzberg	nada84@example.org	321-535-4347	Upravnik	Stracke-Littel	
Julius	Schaden	vmcdermize@example.com	339-446-9275	Poslovnik	Gulgowski Group	
Lindsay	Larson	agheason@example.org	1-559-600-4069	Klijent	-	
Marcelle	Hayes	jones.estra@example.com	1-909-748-0545	Poslovnik	Paucek LLC	
Raquel	Rogahn	ernest95@example.net	220-278-0744	Poslovnik	Lynch and Sons	
Tate	Klocko	hector.brekke@example.net	346.525.4154	Klijent	-	
Zoey	Lowe	reinhold.huels@example.net	+1-279-274-5950	Klijent	-	

Slika 16: Pregled korisnika unutar sistema

Slučaj korištenja 7: Upravljanje korisnicima

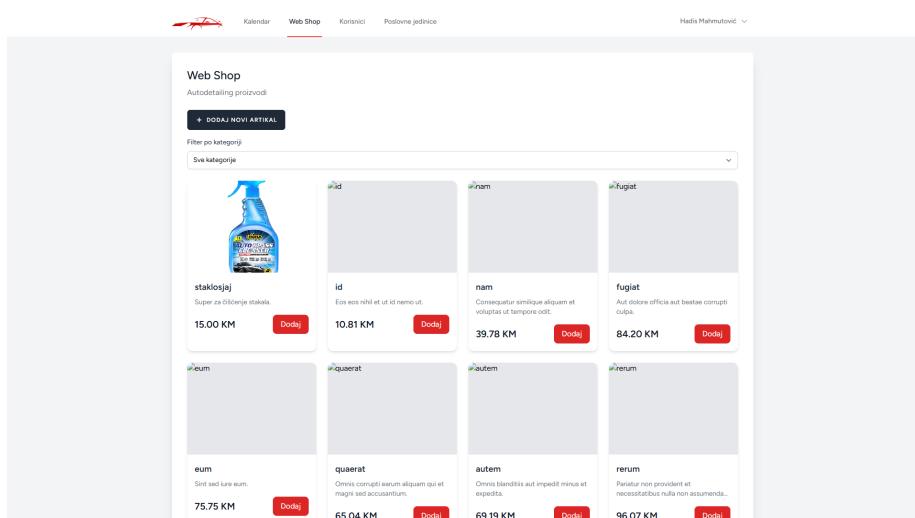
- Naziv SK: Upravljanje korisnicima,
- Akteri SK: Administrator,
- Učesnici SK: Administrator i sistem,
- Preduslov: Administrator je prijavljen na sistem,
- Osnovni scenarij SK:
 - Administrator pristupa sekciji za upravljanje korisnicima,
 - Administrator može dodati, urediti ili obrisati korisnika,
 - Administrator unosi ili mijenja podatke korisnika,
 - Sistem validira unesene podatke,
 - Sistem sprema izmjene u bazu podataka,
 - Sistem prikazuje potvrdu o uspješnoj operaciji.

Slika 17: Uređivanje korisnika od strane administratora

Slučaj korištenja 8: Upravljanje proizvodima

- Naziv SK: Upravljanje proizvodima,
- Akteri SK: Administrator,

- Učesnici SK: Administrator i sistem,
- Preduslov: Administrator je prijavljen na sistem,
- Osnovni scenarij SK:
 - Administrator pristupa sekciji za upravljanje proizvodima,
 - Administrator dodaje novi proizvod ili uređuje postojeći,
 - Administrator unosi ili mijenja podatke o proizvodu,
 - Sistem validira unesene podatke,
 - Sistem sprema podatke u bazu,
 - Sistem prikazuje potvrdu o uspješnoj operaciji.



Slika 18: Pregled proizvoda unutar Web Shop-a

Na osnovu definisanih slučajeva korištenja može se zaključiti da sistem uspješno podržava sve ključne funkcionalnosti potrebne za upravljanje rezervacija, korisnicima i proizvodima. Implementirane funkcionalnosti omogućavaju različitim tipovima korisnika pristup sistemu u skladu sa njihovim privilejima, čime se osigurava kontrolisan i siguran rad aplikacije. Sistem obezbeđuje validaciju unosa podataka, konzistentno čuvanje informacija u bazi podataka, kao i pravovremeno informisanje korisnika o rezultatima izvršenih akcija. Na ovaj način omogućeno je efikasno upravljanje procesom rezervacija, pregled i administracija korisnika, te upravljanje proizvodima, čime sistem ispunjava sve definisane funkcionalne zahtjeve i predstavlja pouzdano rješenje za podršku poslovnim procesima.

5 Zaključak

Razvoj web aplikacije za upravljanje rezervacijama, korisnicima i proizvodima predstavlja uspješnu implementaciju savremenih web tehnologija s ciljem digitalizacije i unapređenja procesa upravljanja poslovnim jedinicama i rezervacijama. Sistem omogućava jednostavno kreiranje i pregled rezervacija, efikasno upravljanje korisnicima i proizvodima, kao i jasno definisanu kontrolu pristupa u skladu sa korisničkim ulogama, uključujući klijenta, uposlenika, poslovodu i administratora. Ovakav pristup doprinosi boljoj organizaciji podataka, povećava efikasnost rada i smanjuje mogućnost grešaka koje su česte kod ručnog vođenja evidencije.

Aplikacija je razvijena korištenjem Laravel PHP frameworka, koji kroz primjenu MVC arhitekture omogućava modularan, pregledan i lako održiv kod. Razdvajanje poslovne logike, korisničkog interfejsa i sloja za pristup podacima omogućilo je jednostavniju implementaciju funkcionalnosti, kao i mogućnost budućih nadogradnji sistema. Za pohranu podataka korišten je MySQL sistem za upravljanje relacijskim bazama podataka, koji obezbjeđuje pouzdano, sigurno i efikasno upravljanje informacija o korisnicima, rezervacijama, proizvodima i poslovnim jedinicama.

U procesu razvoja korišteni su UML dijagrami, uključujući dijagrame slučajeva korištenja, sekvensialne dijagrame i ER dijagram baze podataka, koji su omogućili jasno modeliranje strukture sistema i interakcije između njegovih komponenti. Implementacija REST API-ja omogućila je standardizovanu komunikaciju između klijentskog i serverskog dijela aplikacije, čime je osigurana fleksibilnost i mogućnost integracije sa drugim sistemima ili klijentskim aplikacijama.

Testiranje sistema potvrđilo je ispravnost implementiranih funkcionalnosti, uključujući autentifikaciju korisnika, upravljanje rezervacijama, upravljanje korisnicima i proizvodima, kao i validaciju unesenih podataka. Sistem uspješno obrađuje korisničke zahtjeve, čuva podatke u bazi i prikazuje relevantne informacije korisnicima u skladu sa njihovim ovlaštenjima.

Na osnovu realizovanog projekta može se zaključiti da razvijena web aplikacija predstavlja funkcionalno, stabilno i skalabilno rješenje za upravljanje rezervacijama i pratećim resursima. Sistem pruža dobru osnovu za buduća unapređenja, kao što su implementacija notifikacija, naprednih izvještaja, kalendarskog prikaza rezervacija ili integracija sa eksternim servisima, čime bi se dodatno unaprijedila efikasnost i praktična primjena aplikacije u realnom poslovnom okruženju.

Literatura

- [1] Tutorials Point (Model-View-Controller), Dostupno na linku: <https://www.tutorialspoint.com/mvc/framework/mvc/framework/introduction.htm/>
- [2] Laravel , Dostupno na linku: <https://laravel.com/>
- [3] GeeksforGeeks, Dostupno na linku: <https://www.geeksforgeeks.org/unifiedmodeling-language-uml-introduction/>