Univerzitet u Bihaću
Tehnički fakultet
Odsjek: Elektrotehnika
Smjer: Računarstvo i Informatika

Objektno orijentirane baze podataka

Projektni zadatak

Tema: LABORATORIJA

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

Student:
Aldin Hodžić, 1231

Bihać, januar 2026. godine

Sažetak

U ovom radu razvijena je web aplikacija za laboratoriju s ciljem digitalizacije osnovnih laboratorijskih procesa. Sistem omogućava upravljanje pacijentima, zakazivanje laboratorijskih pretraga, unos i pregled nalaza, te administraciju korisnika i usluga. Aplikacija je realizovana korištenjem Laravel frameworka, uz podršku sigurnog sistema autentifikacije, korisničkih uloga (administrator, laborant i pacijent) i CRUD operacija nad ključnim entitetima. Podaci se čuvaju u relacionoj bazi podataka, dok je komunikacija omogućena putem REST API-ja. Razvijeno rješenje predstavlja funkcionalan i skalabilan sistem za unapređenje rada laboratorije.

Ključne riječi: web aplikacija, laboratorija, Laravel, REST API, baza podataka

Abstract

This paper presents a web-based laboratory application aimed at digitalizing core laboratory processes. The system supports patient management, appointment scheduling, laboratory result handling, and user administration. The application is developed using the Laravel framework and implements secure authentication, role-based access control, and CRUD operations on key entities. Data is stored in a relational database, while system communication is provided through a RESTful API. The developed solution offers a functional and scalable platform for laboratory workflow management.

Keywords: web application, laboratory system, Laravel, REST API, database

Sadržaj

Popis slika

1 Uvod

Razvoj informacionih sistema u oblasti zdravstva i laboratorijske dijagnostike ima ključnu ulogu u unapređenju efikasnosti rada, tačnosti obrade podataka i kvaliteta usluga koje se pružaju pacijentima. Savremene laboratorije sve više napuštaju manuelne i papirne evidencije te prelaze na digitalna rješenja koja omogućavaju centralizovano upravljanje podacima, jednostavniju komunikaciju između osoblja i pacijenata, kao i brži pristup laboratorijskim rezultatima. U tom kontekstu, javlja se potreba za razvojem stabilne, sigurne i funkcionalne web aplikacije koja će objediniti osnovne procese rada jedne laboratorije. Predmet ovog projekta je razvoj web aplikacije za potrebe laboratorije, čiji je cilj automatizacija i digitalizacija ključnih poslovnih procesa, kao što su upravljanje pacijentima, zakazivanje laboratorijskih pretraga, unos i pregled rezultata pretraga, kao i administracija korisnika i šifrarnika laboratorijskih usluga. Aplikacija je namijenjena različitim tipovima korisnika, uključujući administratore, laboratorijsko osoblje (laborante) i pacijente, pri čemu svaki korisnik ima jasno definisana prava i mogućnosti unutar sistema. Aplikacija je razvijena korištenjem **Laravel PHP frameworka**, koji predstavlja jedan od najpopularnijih savremenih alata za razvoj web aplikacija. Laravel omogućava izradu modularnih, sigurnih i skalabilnih sistema, te se zasniva na **MVC (Model–View–Controller)** arhitekturi. Ovakav arhitektonski pristup omogućava jasnu podjelu odgovornosti unutar aplikacije: modeli su zaduženi za rad sa bazom podataka i poslovnim entitetima, kontroleri implementiraju poslovnu logiku i obradu zahtjeva, dok su pogledi (views) odgovorni za prikaz podataka korisniku. Time se postiže veća preglednost koda, lakše održavanje aplikacije i jednostavnija buduća nadogradnja sistema. Za pohranu i upravljanje podacima koristi se **relaciona baza podataka**, koja omogućava strukturirano čuvanje informacija o korisnicima, pacijentima, zakazanim terminima, vrstama laboratorijskih pretraga i rezultatima analiza. Korištenjem **Eloquent ORM-a**, Laravel omogućava rad sa bazom podataka putem objektno orijentisanog pristupa, gdje se zapisi iz baze predstavljaju kao objekti. Ovakav način rada značajno pojednostavljuje manipulaciju podacima, smanjuje potrebu za pisanjem kompleksnih SQL upita i doprinosi većoj sigurnosti i čitljivosti aplikacije. Cilj ovog projekta je izrada funkcionalne, pregledne i pouzdane web aplikacije koja unapređuje rad laboratorije, smanjuje administrativno opterećenje zaposlenih i omogućava pacijentima jednostavniji pristup laboratorijskim uslugama. Korištenjem savremenih tehnologija i principa objektno orijentisanog programiranja, razvijeno rješenje predstavlja kvalitetnu osnovu za dalju nadogradnju i prilagođavanje specifičnim potrebama zdravstvenih ustanova.

2 Modeliranje aplikacije

Modeliranje aplikacije predstavlja jedan od najvažnijih koraka u procesu razvoja softverskog sistema, jer omogućava jasno definisanje strukture aplikacije, njenih funkcionalnosti i odnosa između pojedinih dijelova sistema. Prije same implementacije bilo je neophodno analizirati potrebe laboratorije i korisnika aplikacije, identifikovati ključne procese (upravljanje pacijentima, zakazivanje termina, obrada rezultata pretraga) te definisati način na koji će različite uloge komunicirati sa sistemom. Ovakav pristup omogućava bolje planiranje razvoja, jasniju organizaciju koda i jednostavnije održavanje aplikacije. U okviru projekta laboratorijske web aplikacije, modeliranje obuhvata izradu UML dijagrama koji prikazuju interakciju između korisnika i sistema, tokove aktivnosti, kao i strukturu baze podataka. Posebna pažnja posvećena je definisanju uloga i prava pristupa, s obzirom na to da aplikaciju koriste tri osnovne grupe korisnika: administrator, laborant i pacijent. Administrator ima odgovornost upravljanja korisnicima i šifrarnicima (npr. vrste laboratorijskih pretraga), laborant upravlja zakazivanjima i unosi rezultate, dok pacijent može izvršiti registraciju/prijavu, zakazati pregled te pregledati svoje nalaze. Jasno razdvajanje ovih uloga je ključno kako bi sistem bio siguran i kako bi se osiguralo da svaki korisnik ima pristup samo onim funkcionalnostima koje su mu potrebne. Za prikaz funkcionalnosti sistema korišteni su dijagrami slučajeva upotrebe (Use Case), koji predstavljaju osnovu za razumijevanje zahtjeva i ponašanja sistema iz perspektive korisnika. Na osnovu njih su dalje izrađeni sekvencijalni dijagrami. Pored toga, model baze podataka definisan je kroz prikaz entiteta i njihovih veza (npr. korisnici, pacijenti, zakazivanja, vrste pretraga i rezultati), čime se osigurava konzistentno čuvanje podataka i pravilno povezivanje informacija unutar sistema. Modeliranje je značajno i zbog toga što omogućava rano uočavanje mogućih nedostataka u dizajnu, smanjuje vjerovatnoću grešaka tokom implementacije i olakšava komunikaciju između svih učesnika u razvoju.

2.1 Opis aplikacije

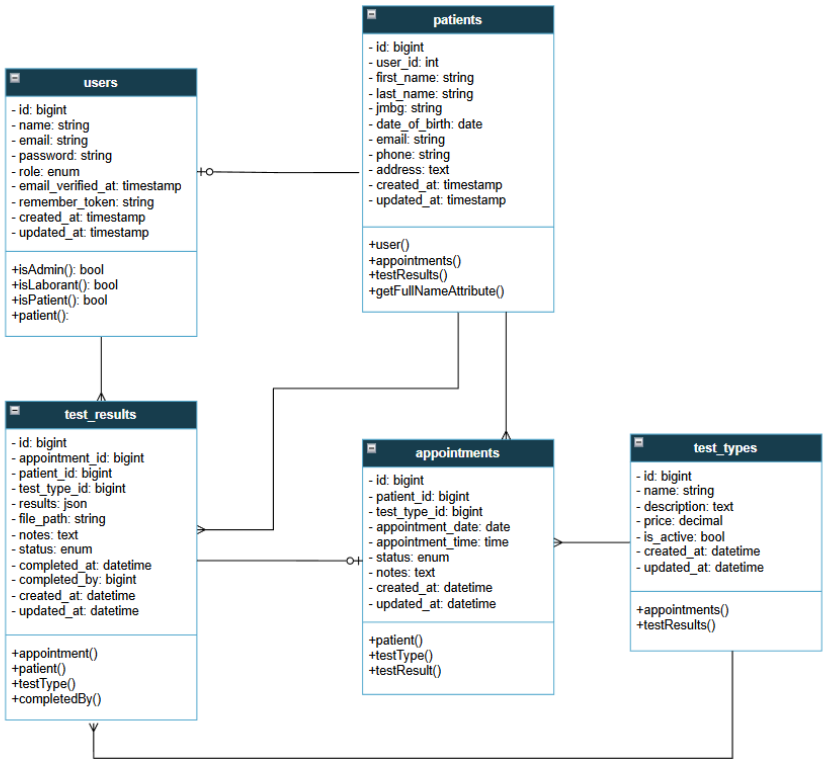
Aplikacija za laboratoriju predstavlja web sistem namijenjen digitalizaciji i unapređenju osnovnih procesa rada laboratorije kroz jednostavno, pregledno i efikasno okruženje. Cilj aplikacije je olakšati upravljanje pacijentima, zakazivanje laboratorijskih pretraga i obradu rezultata, uz jasno definisane uloge korisnika. Pacijentima je omogućena registracija, zakazivanje pretraga i pregled ličnih nalaza, dok laboratorijsko osoblje ima pristup funkcionalnostima za upravljanje terminima i unos rezultata pretraga. Administratori sistema zaduženi su za upravljanje korisnicima i održavanje šifrarnika laboratorijskih usluga. Aplikacija je osmišljena s ciljem da obezbijedi sigurno i intuitivno korištenje, poboljša organizaciju rada laboratorije i omogući pouzdan pristup relevantnim informacijama svim učesnicima sistema.

2.2 Statički UML dijagrami

Statički UML dijagrami predstavljaju vizuelni prikaz strukture softverskog sistema i njegovih osnovnih elemenata. Koriste se za opis arhitekture aplikacije, organizacije podataka i odnosa između pojedinih komponenti sistema. Njihova osnovna svrha je olakšavanje razumijevanja dizajna aplikacije prije same implementacije, kao i pojednostavljivanje održavanja i daljeg razvoja sistema. U procesu razvoja softverskih rješenja, UML dijagrami omogućavaju jasan i sistematičan prikaz strukture aplikacije, međusobnih zavisnosti komponenti i načina na koji su podaci organizovani unutar sistema. Statički dijagrami su fokusirani na prikaz trajnih elemenata sistema, nezavisno od njihovog ponašanja tokom izvršavanja aplikacije. 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 laboratorijske web aplikacije kroz glavne entitete i njihove međusobne odnose. U sistemu se koriste klase **User**, **Patient**, **Appointment**, **TestType** i **TestResult**, koje zajedno omogućavaju upravljanje korisnicima, pacijentima, zakazivanjem laboratorijskih pretraga i obradom rezultata. Klasa *User* predstavlja osnovu za autentifikaciju i definisanje uloga korisnika (administrator, laborant i pacijent), dok je klasa *Patient* povezana sa korisnikom koji ima ulogu pacijenta i sadrži dodatne podatke neophodne za vođenje laboratorijske evidencije. Pacijent može imati više zakazanih termina, dok svaki termin pripada tačno jednom pacijentu i jednoj vrsti laboratorijske pretrage. Klasa *Appointment* povezuje pacijente i vrste pretraga, te sadrži informacije o datumu, statusu i toku obrade pretrage. Svaka vrsta pretrage definisana je klasom *TestType*, koja omogućava centralizovano upravljanje laboratorijskim uslugama. Rezultati pretraga predstavljeni su klasom *TestResult*, pri čemu svaki rezultat pripada jednom zakazanom terminu i unosi ga ovlašteno laboratorijsko osoblje. Ovakva struktura omogućava jasno definisane odnose između entiteta, jednostavno održavanje sistema i njegovu dalju nadogradnju. Klasni dijagram aplikacije prikazan je na slici 1.



Slika 1: Klasni dijagram aplikacije za laboratoriju

2.3 Dinamički UML dijagrami

Dinamički UML dijagrami opisuju ponašanje sistema tokom vremena te prikazuju način na koji se njegove komponente međusobno povezuju i reaguju na određene događaje. Ovi dijagrami omogućavaju jasnije razumijevanje toka izvršavanja procesa, razmjene podataka, kao i interakcije između korisnika i sistema. Posebno su značajni za prikaz stvarnog ponašanja aplikacije tokom izvršavanja pojedinih funkcionalnosti.

Dinamički dijagrami koriste se za modeliranje vremenskih sekvenci, promjena stanja sistema i komunikacije između različitih komponenti. Na taj način pružaju detaljniji uvid u rad aplikacije te pomažu u identifikaciji potencijalnih problema koji se mogu pojaviti tokom faze implementacije. U okviru UML standarda razlikuje se nekoliko vrsta dinamičkih dijagrama:

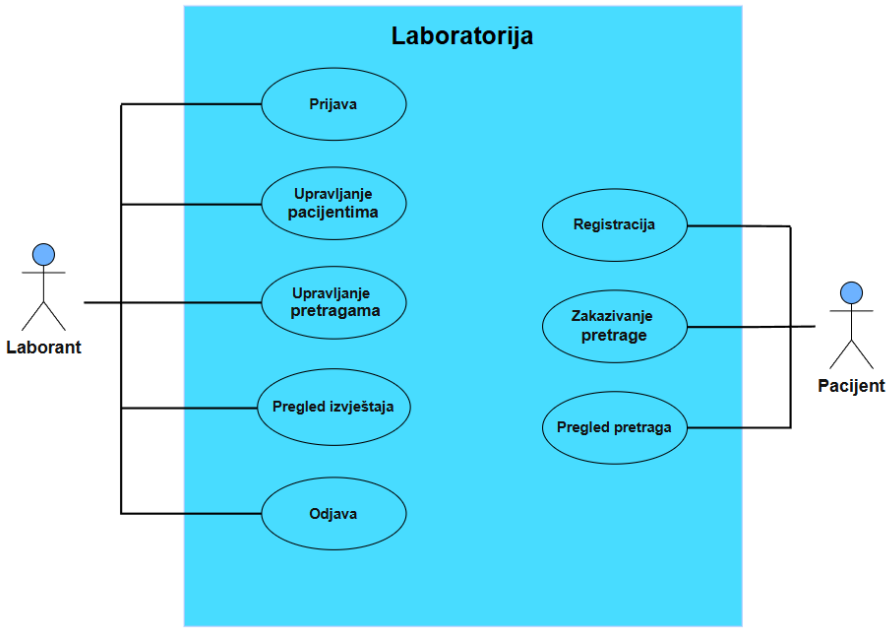
- dijagram aktivnosti,
- dijagram stanja,
- dijagram slučajeva korištenja,
- sekvencijalni dijagram,
- komunikacioni dijagram.

U nastavku rada detaljnije su opisani i prikazani dijagram slučajeva korištenja i sekvencijalni dijagram, jer oni na najjasniji način 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, kao i njihove interakcije sa aplikacijom. U okviru aplikacije za upravljanje laboratorijskim procesima, ovi dijagrami omogućavaju jasan pregled načina na koji različiti tipovi korisnika koriste sistem i koje radnje mogu izvršavati unutar aplikacije. Dijagrami slučajeva korištenjai služe za ilustraciju osnovnih funkcionalnih zahtjeva sistema, bez ulaska u tehničke detalje implementacije. Njihova svrha je prikaz scenarija korištenja sistema, korisničkih uloga i granica sistema, čime se olakšava razumijevanje ponašanja aplikacije u realnim uslovima rada.U aplikaciji laboratorije definisana su dva osnovna tipa aktera: **laborant/admin** i **pacijent**. Laborant, odnosno administrator sistema, ima ovlaštenja za upravljanje podacima o pacijentima, laboratorijskim pretragama, zakazivanjem termina i objavljivanjem laboratorijskih nalaza. Pacijent ima mogućnost registracije i prijave u sistem, zakazivanja laboratorijskih pretraga, kao i pregleda vlastitih nalaza putem aplikacije. Ovakva podjela jasno definiše prava i odgovornosti korisnika unutar sistema te omogućava kontrolisan i siguran rad aplikacije.

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



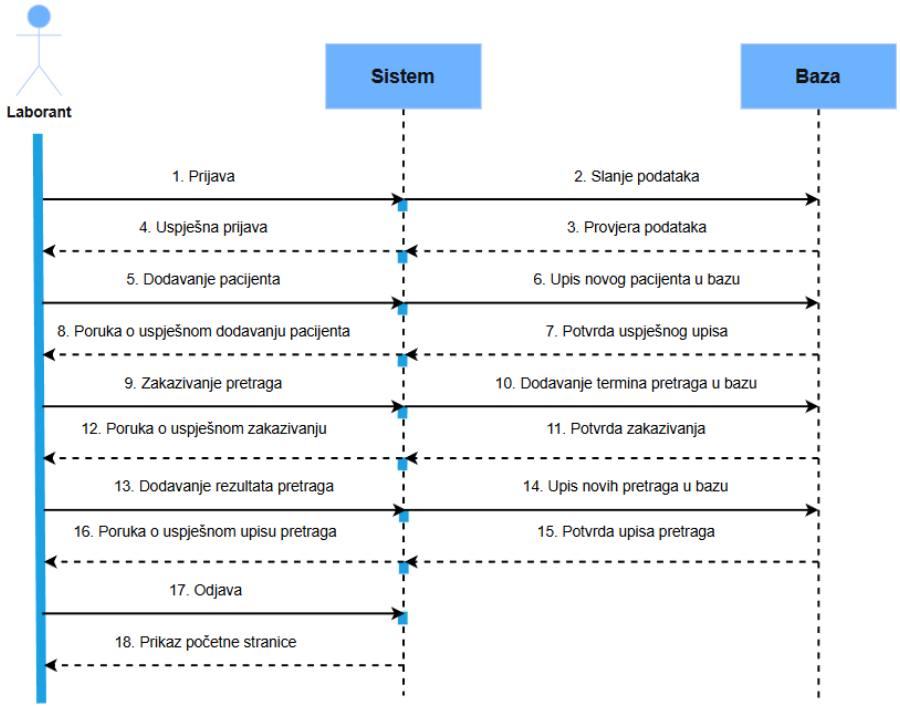
Slika 2: Dijagram slučajeva korištenja

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 dijagram

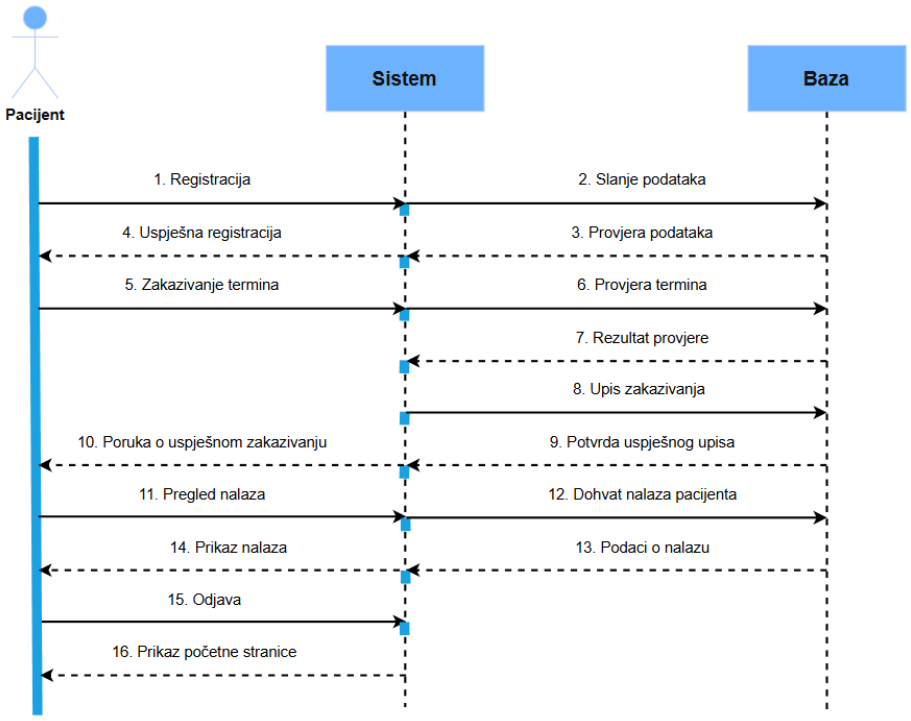
Sekvencijalni dijagram koristi se za prikaz vremenskog slijeda interakcija između aktera i sistema, odnosno načina na koji se razmjenjuju poruke tokom izvršavanja određene funkcionalnosti. Ovaj dijagram omogućava jasno razumijevanje redoslijeda aktivnosti i komunikacije između korisnika, aplikacije i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

Na slici 3 prikazan je sekvencijalni dijagram za laboranta.



Slika 3: Sekvencijalni dijagram za laboranta

Na slici 4 prikazan je sekvencijalni dijagram za registrovanog korisnika. Na slici 4 prikazan je sekvencijalni dijagram za **pacijenta**, koji ilustruje proces prijave u sistem, zakazivanja laboratorijskih pretraga i pregleda vlastitih nalaza. Na ovaj način omogućeno je jasno razumijevanje toka komunikacije između pacijenta i sistema tokom korištenja osnovnih funkcionalnosti namijenjenih krajnjim korisnicima aplikacije.

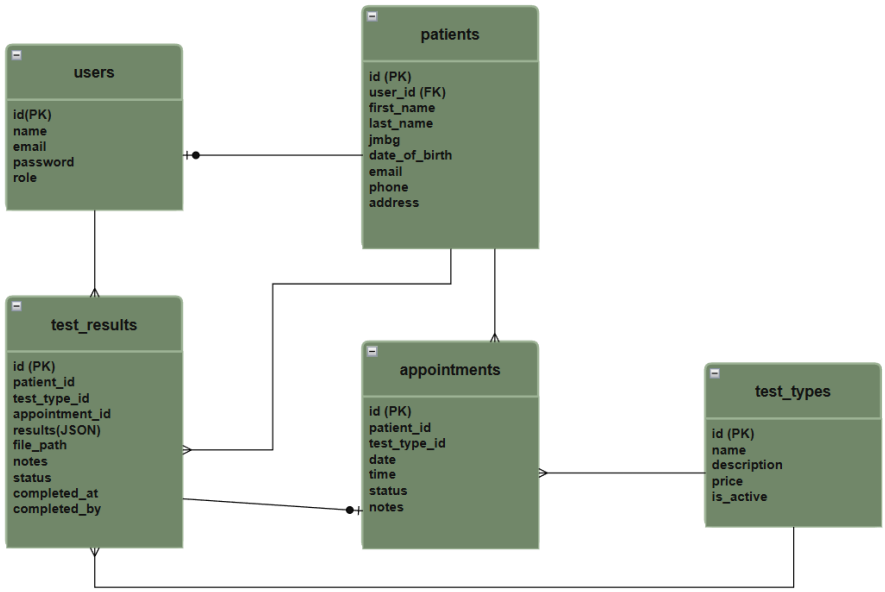


Slika 4: Sekvencijalni dijagram za pacijenta

Prikazani sekvencijalni dijagrami jasno ilustruju razliku u funkcionalnostima između **laboranta** i **pacijenta**, kao i način na koji sistem obrađuje njihove zahtjeve. Kroz ove dijagrame prikazan je tok komunikacije između korisnika, aplikacije i baze podataka tokom izvršavanja ključnih funkcionalnosti. Sekvencijalni dijagrami omogućavaju lakše razumijevanje ponašanja sistema u realnim uslovima rada te predstavljaju važan dio dokumentacije aplikacije, jer na jasan i pregledan način opisuju dinamiku izvršavanja procesa unutar sistema.

2.4 ER dijagram baze podataka

Baza podataka sistema opisana je pomoću ER (*Entity Relationship*) dijagrama, koji prikazuje osnovne entitete i njihove međusobne veze. Sistem aplikacije za upravljanje laboratorijskim procesima sastoji se od **pet glavnih tabela**, koje omogućavaju efikasno vođenje podataka o korisnicima, pacijentima, laboratorijskim pretragama i rezultatima. Entitet **User** predstavlja registrovane korisnike sistema, koji mogu imati ulogu **laboranta/admina** ili **pacijenta**. Ovaj entitet sadrži osnovne podatke potrebne za autentifikaciju i autorizaciju korisnika. Entitet **Patient** sadrži detaljne informacije o pacijentima laboratorije i povezan je sa korisničkim nalogom u slučaju da pacijent ima pristup aplikaciji. Entitet **Appointment** predstavlja zakazane laboratorijske pretrage i povezuje pacijente sa odgovarajućim terminima i vrstama pretraga. Entitet **TestType** sadrži šifarnik laboratorijskih pretraga i omogućava standardizovano definisanje dostupnih testova. Entitet **TestResult** čuva rezultate izvršenih laboratorijskih analiza i povezan je sa pacijentima i zakazanim terminima. Ovako definisana struktura baze podataka omogućava jasno modeliranje poslovne logike sistema, održavanje integriteta podataka i efikasnu implementaciju funkcionalnosti aplikacije.



Slika 5: Entity Relationship dijagram

Sistem koristi pretežno relacije tipa 1:N. Korisnik može biti povezan sa više pacijenata i laboratorijskih nalaza, pacijent može imati više zakazivanja i rezultata, dok se svaki laboratorijski nalaz odnosi na jednog pacijenta, jedno zakazivanje i jedan tip testa.

3 Implementacija

3.1 Tehnologija izrade aplikacije

Aplikacija laboratorije razvijena je korištenjem programskog jezika **PHP** uz primjenu **Laravel frameworka**, koji predstavlja jedan od najpopularnijih alata za razvoj modernih web aplikacija. Laravel je zasnovan na **M**VC arhitekturi, koja omogućava jasno razdvajanje poslovne logike, korisničkog interfejsa i upravljanja podacima, čime se postiže bolja organizacija koda i lakše održavanje sistema. Primjena MVC arhitekture omogućava veću preglednost aplikacije, jednostavnije testiranje, kao i lakšu nadogradnju i proširenje funkcionalnosti. **Modeli** su zaduženi za rad s podacima i komunikaciju s bazom podataka, **kontroleri** obrađuju korisničke zahtjeve i upravljaju poslovnom logikom sistema, dok su **prikazi (views)** odgovorni za vizuelnu prezentaciju informacija korisnicima, uključujući pacijente i laboratorijsko osoblje. 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 podataka, što je posebno pogodno za aplikacije koje upravljaju većim brojem međusobno povezanih entiteta, kao što su korisnici, pacijenti, zakazivanja, laboratorijski testovi i rezultati pretraga.

3.2 Laravel

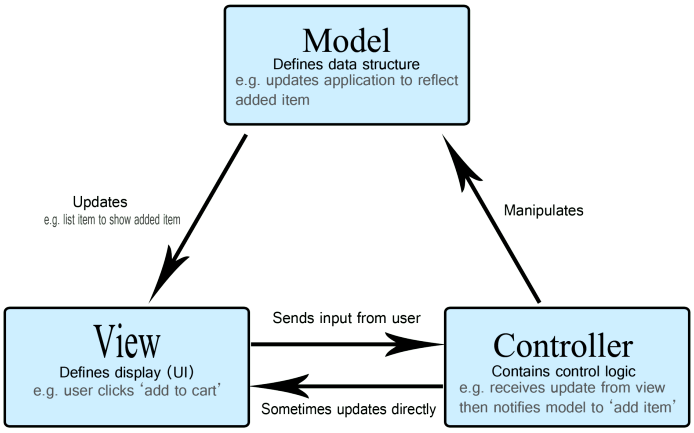
Laravel je *open-source* PHP framework namijenjen razvoju modernih, sigurnih i skalabilnih web aplikacija. Odlikuje se jednostavnom i čitljivom sintaksom, jasno definisanom strukturom projekta te bogatim skupom ugrađenih funkcionalnosti koje značajno ubrzavaju proces razvoja aplikacije. Jedna od ključnih prednosti Laravel frameworka jeste implementacija **objektno–relacionog mapiranja (ORM)** putem **Eloquent ORM-a**, koji omogućava intuitivnu, sigurnu i efikasnu komunikaciju sa bazom podataka. Na ovaj način pojednostavljen je rad sa relacionim podacima koji su od posebne važnosti u laboratorijskom informacionom sistemu, kao što su pacijenti, zakazivanja, laboratorijski testovi i rezultati pretraga. Laravel pruža napredne mehanizme za upravljanje **rutama**, **autentifikacijom i autorizacijom korisnika**, **validacijom podataka** i **sigurnošću aplikacije**, čime se osigurava pouzdan rad sistema i zaštita osjetljivih medicinskih podataka. Također, framework nudi podršku za razvoj modularnih komponenti i korištenje **Blade templating sistema**, koji omogućava jednostavno i efikasno kreiranje dinamičkih korisničkih interfejsa prilagođenih različitim ulogama korisnika, kao što su pacijenti i laboratorijsko osoblje.

3.3 MySQL

MySQL je relacioni sistem za upravljanje bazama podataka koji omogućava efikasno čuvanje, obradu i upravljanje podacima. Podaci su organizovani u tabele koje se sastoje od redova i kolona, pri čemu svaka tabela predstavlja određeni entitet unutar sistema. Ovakav pristup omogućava jasno definisane odnose između podataka i efikasno upravljanje složenim strukturama informacija. Zahvaljujući svojoj pouzdanosti, visokoj brzini izvršavanja i širokoj primjeni, MySQL predstavlja jedan od najčešće korištenih sistema za upravljanje bazama podataka u web aplikacijama. Njegova stabilnost i podrška za rad sa velikim količinama podataka čine ga pogodnim izborom za razvoj informacionih sistema koji zahtijevaju visok nivo tačnosti i sigurnosti.

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. Ovakav pristup omogućava bolju organizaciju koda, lakše održavanje te jednostavnije proširivanje i nadogradnju 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 siguran i kontrolisan pristup podacima putem jasno definisanih metoda. U kontekstu aplikacije laboratorije, modeli obuhvataju entitete kao što su korisnici, pacijenti, zakazivanja, tipovi laboratorijskih testova i rezultati pretraga. 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 kao što su forme za registraciju i zakazivanje, tabele sa listama termina, kao i prikaz laboratorijskih nalaza. Jedan model može imati više različitih pogleda, čime se omogućava fleksibilan i prilagodljiv prikaz sadržaja u zavisnosti od uloge korisnika, poput pacijenata ili laboratorijskog osoblja. 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 ili odgovor prikazati korisniku. Na ovaj način kontroler upravlja tokom rada aplikacije i osigurava pravilnu komunikaciju između svih komponenti sistema.



Slika 6: MVC arhitektura

Kao što se može vidjeti na slici 6, prikazana je MVC arhitektura te komunikacija izmedu komponenti, tačnije modela, pogleda i kontrolera.

3.5 Objektno-relaciono mapiranje

Objektno-relaciono mapiranje (**ORM – Object Relational Mapping**) predstavlja mehanizam koji omogućava rad sa bazom podataka kroz objekte i klase unutar aplikacije, bez potrebe za direktnim pisanjem SQL upita. ORM omogućava povezivanje objekata iz programskog jezika sa tabelama u bazi podataka, čime se pojednostavljuje upravljanje podacima i unapređuje čitljivost i održivost koda. Korištenjem ORM pristupa, programeri mogu izvršavati osnovne operacije nad podacima, kao što su dodavanje, ažuriranje, brisanje i čitanje zapisa, pomoću metoda i objekata, dok se generisanje odgovarajućih SQL upita odvija automatski u pozadini. Na ovaj način smanjuje se mogućnost grešaka koje se često javljaju prilikom ručnog pisanja SQL upita i olakšava se rad sa bazom podataka. ORM također omogućava lakše testiranje poslovne logike aplikacije, jer aplikacijski kod nije direktno vezan 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.

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 bezstanjskog (*stateless*) rada, pri čemu svaki resurs unutar sistema ima jedinstveni identifikator (**URI – Uniform Resource Identifier**). U kontekstu aplikacije laboratorije, resursi REST API-ja obuhvataju entitete kao što su korisnici, pacijenti, zakazivanja, tipovi laboratorijskih testova i rezultati pretraga. Klijent komunicira sa serverom koristeći standardne HTTP metode, kao što su **GET**, **POST**, **PUT** i **DELETE**, koje omogućavaju dohvat, kreiranje, ažuriranje i brisanje podataka. Razmjena podataka se najčešće vrši u **JSON** formatu, što omogućava jednostavnu obradu, dobru čitljivost i kompatibilnost sa različitim platformama i tehnologijama. Primjena REST API arhitekture omogućava modularan i fleksibilan dizajn sistema, gdje su funkcionalnosti jasno razdvojene i lako proširive. Ovakav pristup olakšava održavanje aplikacije, povećava njenu skalabilnost i omogućava jednostavnu integraciju sa drugim servisima ili klijentskim aplikacijama, uključujući web i mobilne platforme.

4 Analiza rada aplikacije

Analiza rada web aplikacije provedena je kroz razmatranje ključnih slučajeva korištenja, pri čemu su definisani akteri sistema, opisani osnovni tokovi procesa, kao i moguće reakcije sistema u zavisnosti od ishoda izvršenja pojedinih aktivnosti.

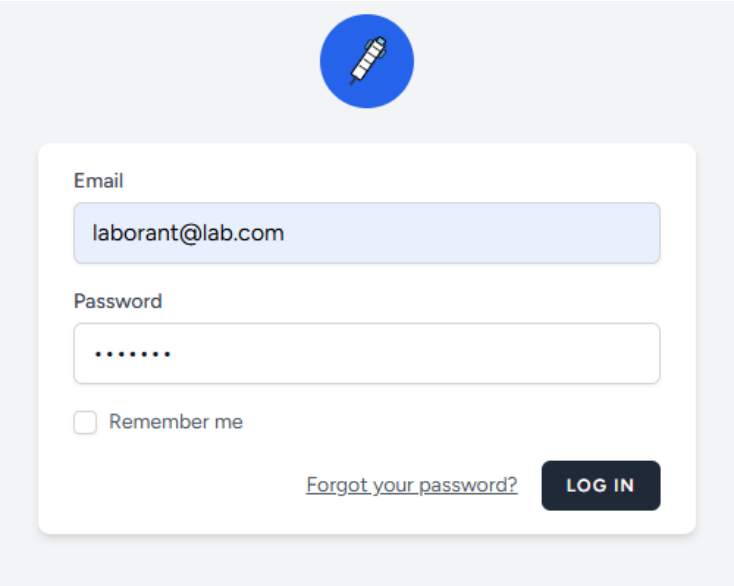
4.1 Opis slučajeva korištenja za laboranta

Slučaj korištenja 1: Prijava na sistem

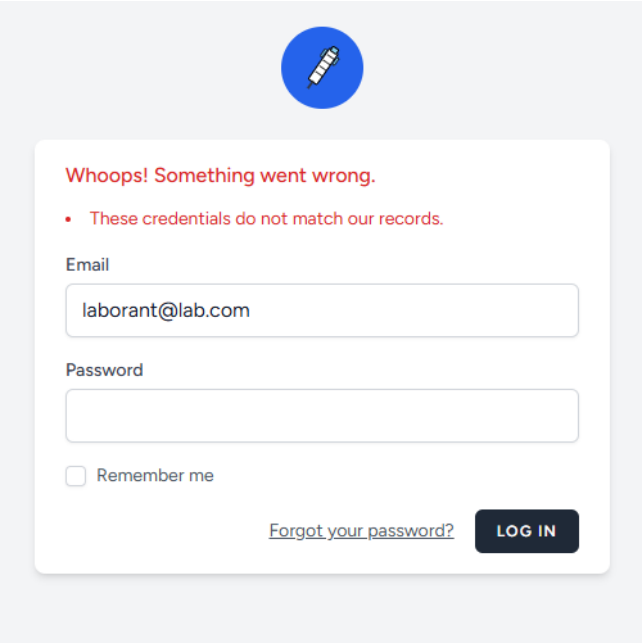
- Naziv SK: Prijava na sistem,
- Akteri SK: Registrovani korisnik(laborant ili pacijent),
- Učesnici SK: Akter i sistem,
- Preduslov: Postoji registrovan korisnik sa validnim podacima,
- Osnovni scenarij SK:
  - Akter unosi e-mail adresu i lozinku,
  - Sistem provjerava ispravnost unesenih podataka,
  - Sistem omogućava pristup korisničkom dijelu aplikacije.

Forma za prijavu prikazana je na slici 7.

- Sporedni scenarij SK:
  - Uneseni podaci nisu ispravni,
  - Ispisuje se poruka prikazana na slici 8.



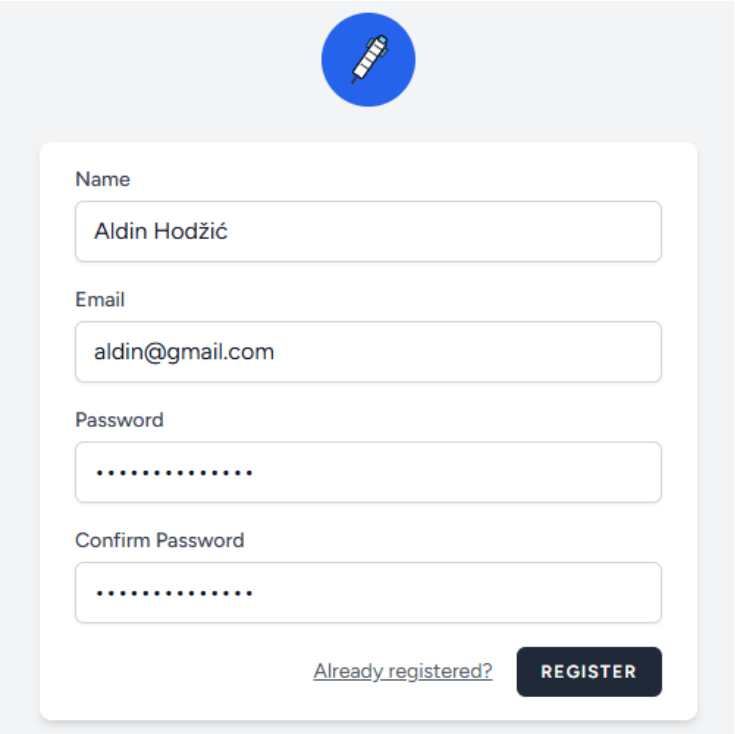
Slika 7: Prijava na sistem



Slika 8: Prijava na sistem netačna

Slučaj korištenja 2: Registracija pacijenta

- Naziv SK: Registracija pacijenta,
- Akteri SK: Neregistrovani korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Korisnik nije registrovan na sistem,
- Osnovni scenarij SK:
  - Akter unosi ime, e-mail adresu i lozinku,
  - Sistem validira unesene podatke,
  - Sistem kreira novi korisnički račun,
  - Sistem obavještava korisnika o uspješnoj registraciji.



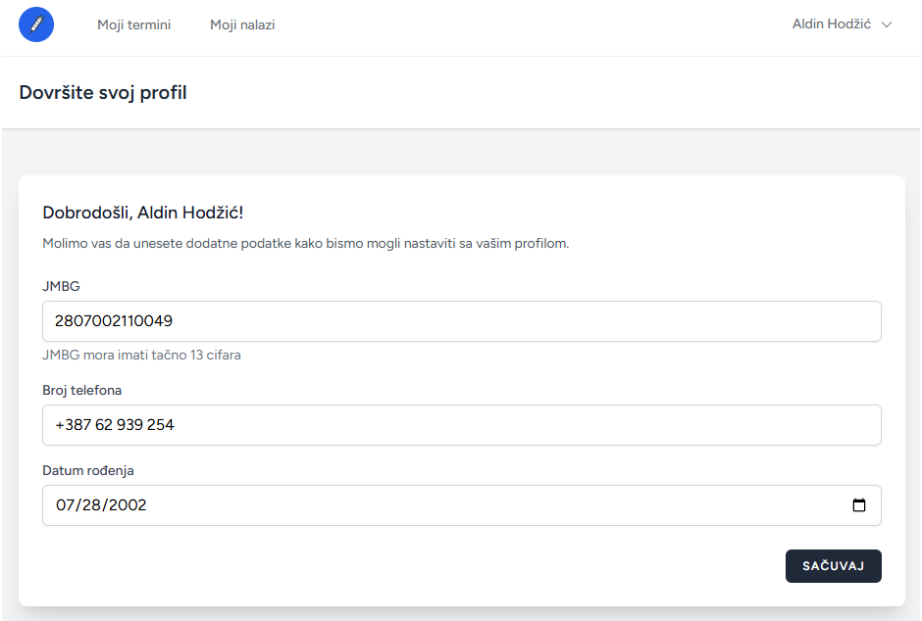
Slika 9: Registracija novog pacijenta

Slučaj korištenja 3: Dovršavanje profila pacijenta

- Naziv SK: Dovršavanje profila pacijenta,

- Akteri SK: Pacijent
- Učesnici SK: Akter i sistem,
- Preduslov: Pacijent je prijavljen i profil nije dovršen (nedostaje JMBG/telefon/datum rođenja),
- Osnovni scenarij SK:
  - Akter otvara formu za dovršavanje profila,
  - Akter unosi JMBG, telefon i datum rođenja ,
  - Sistem validira podatke i snima izmjene profila .

Stranica za unos potrebnih podataka za dovršavanje registracije pacijenta predstavljena je na slici 10.



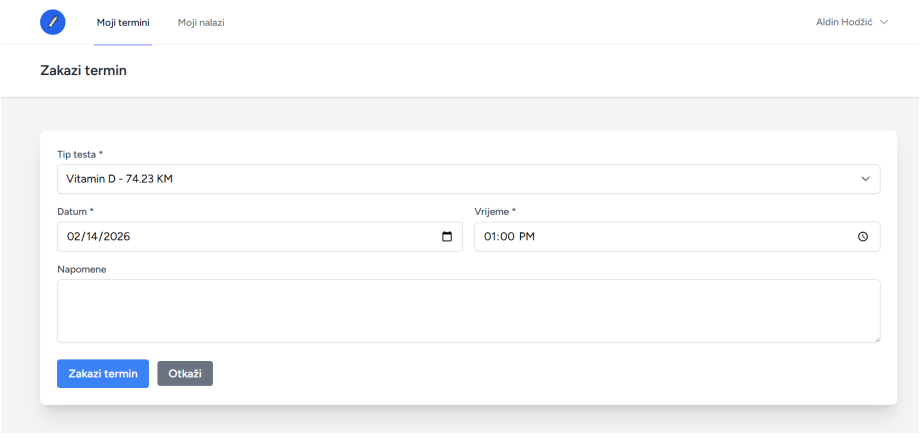
Slika 10: Dovršavanje registracije

Slučaj korištenja 4: Zakazivanje termina (pacijent)

- Naziv SK: Zakazivanje termina,
- Akteri SK: Pacijent
- Učesnici SK: Akter i sistem,
- Preduslov: Pacijent je prijavljen i ima dovršen profil; u sistemu postoje tipovi testova,

- Osnovni scenarij SK:
  - Akter bira opciju ”Zakaži termin”,
  - Akter odabire tip testa i unosi datum/vrijeme termina ,
  - Sistem validira podatke i kreira termin sa statusom ”scheduled” .
  - Sistem prikazuje potvrdu i listu pacijentovih termina .

Stranica za unos potrebnih podataka za dovršavanje registracije pacijenta predstavljena je na slici 11.

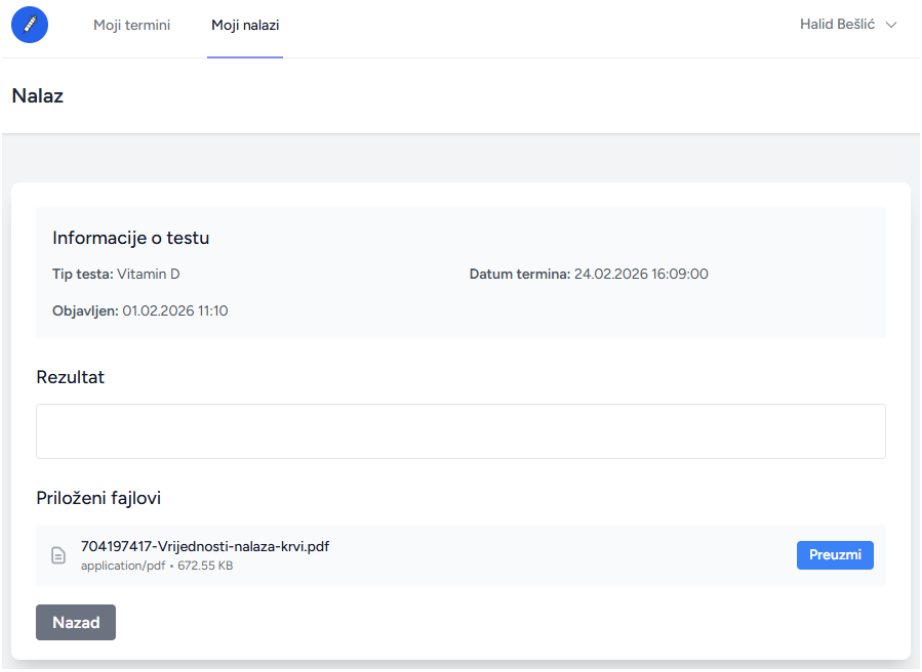


Slika 11: Stranica za zakazivanje termina

Slučaj korištenja 5: Pregled nalaza (pacijent)

- Naziv SK: Pregled nalaza,
- Akteri SK: Pacijent,
- Učesnici SK: Akter i sistem,
- Preduslov: Pacijent je prijavljen,
- Osnovni scenarij SK:
  - Akter otvara opciju ”Moji nalazi”,
  - Sistem prikazuje listu nalaza pacijenata koji su objavljeni,
  - Akter bira nalaz i sistem prikazuje detalje (tekst ili PDF),

Stranica sa detaljima oglasa za posao prikazana je na slici 12.

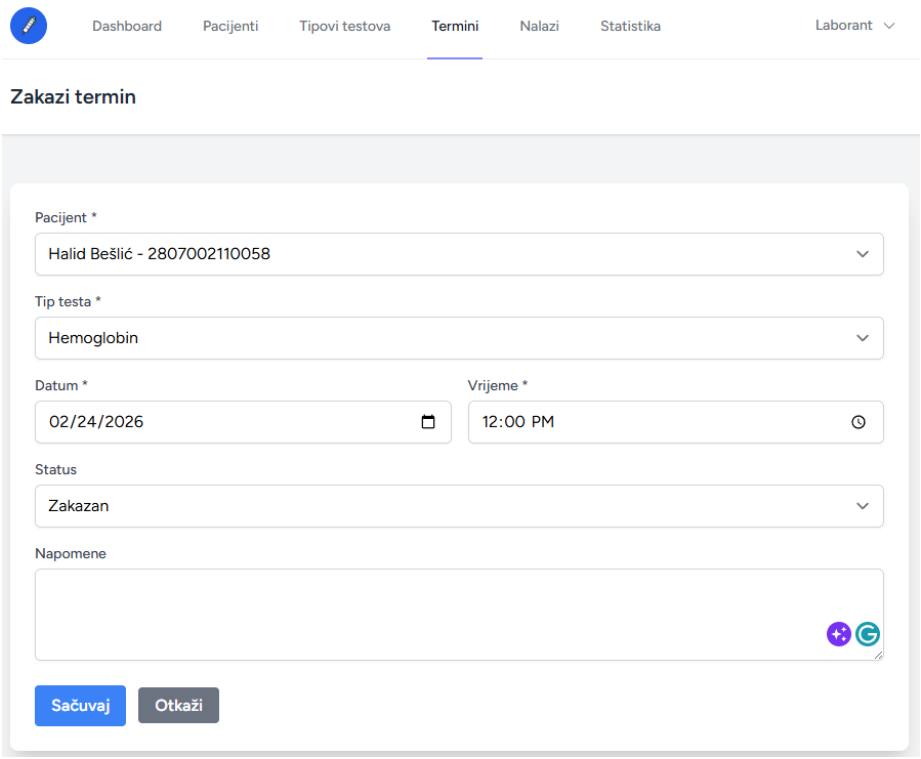


Slika 12: Prikaz nalaza pacijenta

Slučaj korištenja 6: Unos nalaza (laborant)

- Naziv SK: Unos nalaza,
- Akteri SK: Laborant,
- Učesnici SK: Akter i sistem,
- Preduslov: Laborant je prijavljen, postoji termin bez dodanog nalaza,
- Osnovni scenarij SK:
  - Akter otvara sekciju ”Nalazi” i bira opciju ”Novi nalaz”,
  - Sistem pokazuje listu termina koji nemaju nalaz,
  - Akter bira termin i dodaje nalaz,

Stranica za unos nalaza pacijenta prikazana je na slici 13.

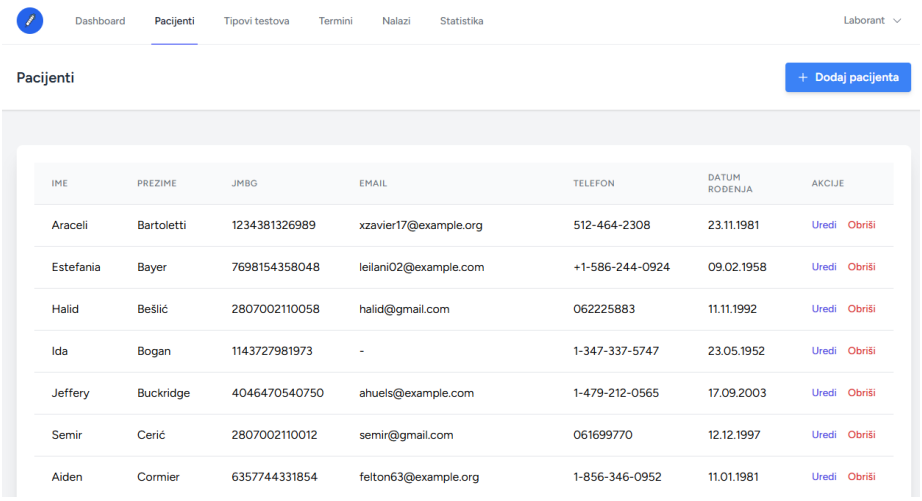


Slika 13: Unos nalaza pacijenta

Slučaj korištenja 7: Pregled pacijenata (laborant)

- Naziv SK: Pregled pacijenata,
- Akteri SK: Laborant,
- Učesnici SK: Akter i sistem,
- Preduslov: Laborant je prijavljen u sistem,
- Osnovni scenarij SK:
  - Akter otvara sekciju ”Pacijenti” u navigaciji,
  - Sistem pokazuje listu svih pacijenata u sistemu,
  - Akter bira određenog pacijenta iz liste,

Stranica za pregled pacijenta prikazana je na slici 14.

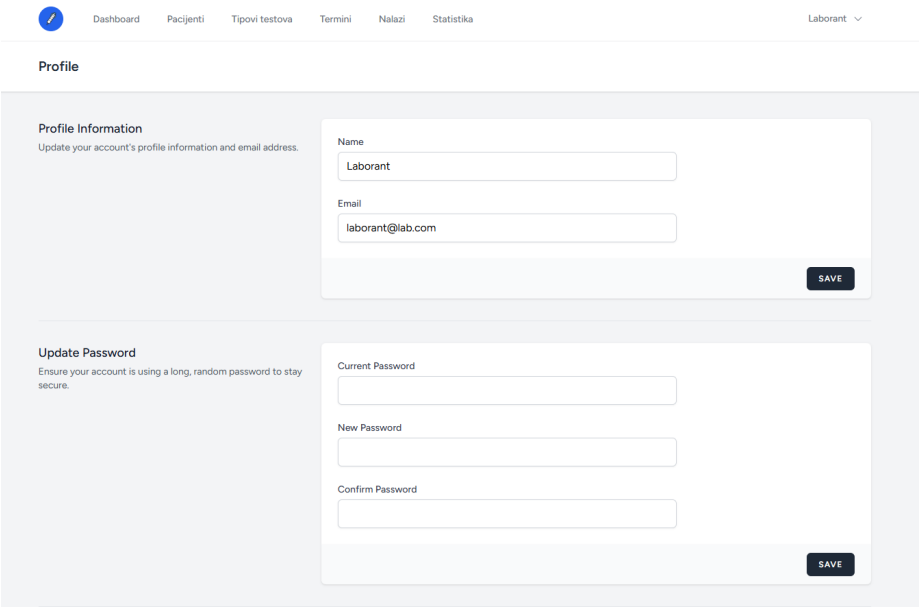


Slika 14: Stranica za pregled pacijenata

Slučaj korištenja 8: Ažuriranje profila (laborant)

- Naziv SK:Ažuriranje profila,
- Akteri SK: Laborant,
- Učesnici SK: Akter i sistem,
- Preduslov: Laborant je prijavljen na sistem,
- Osnovni scenarij SK:
  - Akter otvara opciju „Moj profil“ ,
  - Sistem prikazuje trenutne podatke profila laboranta ,
  - Akter mijenja željene podatke i snima izmjene ,
  - Sistem prikazuje poruku o uspješnom ažuriranju profila.

Stranica za ažuriranje profila prikazana je na slici 15.



Slika 15: Stranica za ažuriranje profila

Slučaj korištenja 9: Pregled dashboard-a (laborant)

- Naziv SK:Pregled dashboard-a,
- Akteri SK: Laborant,
- Učesnici SK: Akter i sistem,
- Preduslov: Laborant je prijavljen na sistem,
- Osnovni scenarij SK:

  - Akter se prijavljuje na sistem ,

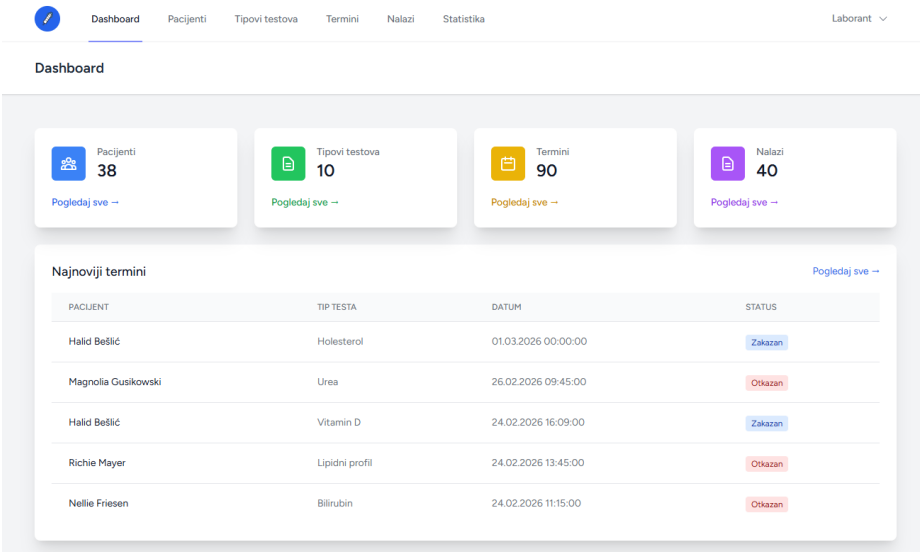
  - Sistem automatski preusmjerava aktera na početnu stranicu (Dashboard) ,

  - Sistem prikazuje sažeti pregled ključnih informacija (broj pacijenata, zakazanih termina, nalaza i tipova testova) ,

  - Sistem prikazuje tabelu na najnovijim terminima.

  - Akter može kliknuti na opciju „Pogledaj sve“ kako bi pristupio detaljnom prikazu pacijenata, tipova testova, termina ili nalaza.

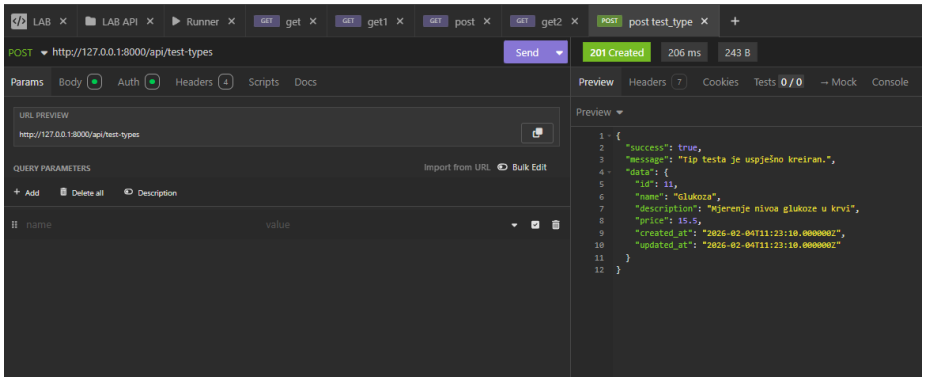
Stranica za prikaz dashboard-a prikazana je na slici 16.



Slika 16: Stranica za ažuriranje profila

4.2 Testiranje API-a

U ovom dijelu rada opisan je način testiranja REST API-ja laboratorijske web aplikacije. Za testiranje je korišten alat **Insomnia**, koji omogućava jednostavno slanje HTTP zahtjeva i pregled odgovora API-ja u **JSON** formatu. Tokom testiranja provjeravana je ispravnost rada API-ja, s posebnim fokusom na HTTP status kodove, validaciju ulaznih podataka i strukturu vraćenih odgovora. Testiranje je obuhvatilo više API ruta, pri čemu je posebna pažnja posvećena provjeri funkcionalnosti za kreiranje novih zapisa putem POST zahtjeva. Kao primjer testiranja, korišten je **POST zahtjev za dodavanje novog tipa nalaza**, koji omogućava administrativnom ili laboratorijskom osoblju unos novih vrsta pretraga u sistem. Prilikom slanja POST zahtjeva putem alata Insomnia, sistem je primao podatke o nazivu testa, opisu i cijeni, vršio validaciju unesenih vrijednosti, te u slučaju ispravnog unosa vraćao JSON odgovor sa statusom uspješnog izvršenja i podacima o novokreiranom tipu testa. U slučaju neispravnih ili nepotpunih podataka, API je vraćao odgovarajuću poruku o grešci i HTTP status kod koji ukazuje na problem sa validacijom. Na slici je prikazan primjer testiranja navedene API rute u alatu Insomnia, gdje se vidi slanje POST zahtjeva za dodavanje novog tipa testa, kao i povratni JSON odgovor servera. Ovakav način testiranja omogućava provjeru ispravnosti rada API-ja i potvrđuje da sistem pravilno obrađuje različite scenarije unosa podataka. Testni zahtjevi korišteni su isključivo u razvojnom okruženju i ne koriste se u produkcionom sistemu.
Prikaz POST zahtjeva za dodavanje tipa nalaza prikazan je na slici 17.



Slika 17: Zahtjev za dodavanje tipa nalaza

5 Zaključak

U okviru ovog rada razvijena je web aplikacija za potrebe laboratorije, čiji je cilj bio digitalizacija i unapređenje osnovnih poslovnih procesa, uključujući upravljanje pacijentima, zakazivanje laboratorijskih pretraga, obradu i pregled rezultata, kao i administraciju korisnika i šifrarnika laboratorijskih usluga. Aplikacija je realizovana korištenjem Laravel frameworka i zasnovana je na savremenim principima razvoja web aplikacija, uz primjenu REST API arhitekture i objektno orijentisanog pristupa radu sa bazom podataka. Tokom izrade aplikacije provedena je detaljna analiza zahtjeva sistema, na osnovu koje su definisani akteri i slučajevi korištenja, a zatim izrađeni odgovarajući UML dijagrami koji opisuju strukturu i ponašanje sistema. Implementirane su migracije baze podataka za sve ključne entitete, uz pripadajuće modele, relacije, seedere i factories, čime je omogućen konzistentan i jednostavan rad sa podacima u razvojnom okruženju. Posebna pažnja posvećena je implementaciji autentifikacije i autorizacije korisnika, pri čemu su jasno razdvojene uloge administratora, laboranta i pacijenta, a pristup funkcionalnostima ograničen u skladu s njihovim ovlaštenjima. Aplikacija koristi REST API za komunikaciju između klijenta i servera, pri čemu su implementirane osnovne CRUD operacije nad odabranim entitetima, kao i specijalizovane API rute za zakazivanje termina, unos i objavu laboratorijskih nalaza. Također je omogućen rad sa fajlovima, uključujući upload i preuzimanje PDF dokumenata laboratorijskih rezultata, uz odgovarajuću validaciju i sigurnosne mehanizme. Kroz implementaciju agregiranih upita i spajanje podataka iz više tabela realizovan je modul za statistiku, koji pruža uvid u rad laboratorije i olakšava donošenje odluka. Testiranje aplikacije izvršeno je korištenjem alata Insomnia, gdje su provjeravani različiti API scenariji, uključujući uspješne i neuspješne zahtjeve, validaciju podataka i ispravnost HTTP status kodova. Na taj način potvrđeno je da API pravilno obrađuje zahtjeve i vraća očekivane odgovore u JSON formatu. Projekat je dostupan putem GitHub repozitorija, uz prateću tehničku dokumentaciju koja opisuje arhitekturu sistema, API rute i način pokretanja aplikacije. Rezultat ovog rada je funkcionalna, pregledna i sigurna laboratorijska web aplikacija koja zadovoljava sve postavljene zahtjeve i predstavlja kvalitetnu osnovu za dalju nadogradnju. Sistem je dizajniran na način koji omogućava lako proširenje funkcionalnosti, integraciju sa eksternim servisima i prilagođavanje specifičnim potrebama zdravstvenih ustanova, čime se potvrđuje praktična primjena savremenih tehnologija u razvoju informacionih sistema.

Literatura

[1] Laravel , Dostupno na linku: https://laravel.com/