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

# Objektno orijentirane baze podataka

Projektni zadatak

Tema: Online upis studenata na unbi

**Profesor:** Prof.dr Admir Midžić
**Asistent:** Zinaid Kapić MA ing. el.

**Student:**
Eldar Džanić, 1251

Bihać, februar 2026. godine

# Sažetak

U radu je prikazan razvoj i implementacija web aplikacije za online prijavu i obradu upisa studenata na Univerzitet u Bihaću. Aplikacija je razvijena korištenjem PHP programskog jezika uz primjenu Laravel frameworka, Jetstream paketa i Livewire komponenti. Ovakav pristup omogućava jasno razdvajanje poslovne logike, korisničkog interfejsa i upravljanja podacima kroz MVC (Model–View–Controller) arhitekturu, čime se postiže preglednost, modularnost i lakše održavanje sistema. Za rad sa bazom podataka korišten je MySQL. Podaci o korisnicima, fakultetima, odsjecima, predmetima, prijavama, ocjenama, dokumentima i administrativnim napomenama organizovani su u medusobno povezane tabele putem primarnih i stranih ključeva. Objektno-relaciono mapiranje realizovano je putem Eloquent ORM-a, čime je omogućen rad sa podacima u obliku objekata. Implementirane su dvije korisničke uloge: kandidat za upis (applicant) i administrator (admin). Kandidati mogu kreirati prijavu, birati fakultet i odsjek, unositi ili ručno ispravljati ocjene iz relevantnih predmeta, uploadovati dokumentnu evidenciju (svjedodžbu, ličnu kartu, rodni list, dokaz uplate) te koristiti OCR za automatsko prepoznavanje ocjena iz PDF dokumenata ili skeniranih slika. Bodovi se automatski izračunavaju prema pravilima bodovanja definisanim za svaki odsjek. Administrator upravlja prijavama, pregledava dokumente i bodove, mijenja status prijave (Draft, Submitted, Under review, Accepted, Rejected, Needs correction), dodaje napomene i može exportovati podatke u CSV formatu. Implementirane su migracije baze podataka, seederi za test podatke te Laravel Storage za čuvanje dokumenata.



**Ključne riječi**: Web aplikacija, Laravel, Jetstream, Livewire, PHP, MySQL, MVC arhitektura, ORM, OCR



# Abstract

This paper presents the development and implementation of a web application for online student enrollment and application processing at the University of Bihać. The application was developed using PHP and the Laravel framework with Jetstream and Livewire components. This approach enables a clear separation of business logic, user interface, and data management through the Model–View–Controller (MVC) architecture, resulting in better organization, modularity, and easier maintenance. MySQL was used as the relational database management system. Data related to users, faculties, departments, subjects, applications, grades, documents, and administrative notes are stored in interconnected tables using primary and foreign keys. Object-relational mapping was implemented through Laravel's Eloquent ORM, allowing database operations to be performed in an object-oriented manner. Two user roles have been implemented: applicant and admin. Applicants can create an application, select faculty and department, enter or manually correct grades for relevant subjects, upload

required documents (transcript, ID card, birth certificate, payment proof), and use OCR for automatic grade extraction from PDF documents or scanned images. Points are calculated automatically according to scoring rules defined for each department. Administrators manage applications, review documents and scores, change application status (Draft, Submitted, Under review, Accepted, Rejected, Needs correction), add notes, and can export data to CSV format. Database migrations, seeders for test data, and Laravel Storage for document storage have been implemented.

**Keywords**: Web application, Laravel, Jetstream, Livewire, PHP, MySQL, MVC architecture, ORM, OCR

# Sadržaj

# Popis slika

# 1 Uvod

Savremeno društvo karakteriše intenzivan razvoj informacionih tehnologija i sve veća digitalizacija svakodnevnih aktivnosti. Web aplikacije postale su neizostavan dio poslovnih sistema, obrazovanja, administracije i javnih institucija. Digitalna rješenja omogućavaju efikasnije upravljanje podacima, automatizaciju procesa i jednostavniju komunikaciju između korisnika i sistema. Posebno su važne aplikacije koje olakšavaju administrativne tokove, smanjuju papirnu dokumentaciju i ubrzavaju obradu podataka. U kontekstu visokog obrazovanja, tradicionalan način vodenja upisa studenata — ručni prikupljanje dokumentata, evidencija prijava i obračun bodova — često dovoditi do grešaka, kašnjenja i otežane organizacije. Razvoj web aplikacije za online upis predstavlja rješenje koje omogućava centralizovano upravljanje prijavama, transparentnost procesa i bolju kontrolu nad tokom prijave i selekcije kandidata. Predmet ovog rada je razvoj i implementacija web aplikacije za online prijavu i obradu upisa studenata na Univerzitet u Bihaću. Aplikacija omogućava kandidatima registraciju i prijavu na sistem, odabir fakulteta i odsjeka, unos ocjena iz relevantnih predmeta, upload dokumenata, automatsko prepoznavanje ocjena iz dokumenata (OCR) te praćenje statusa vlastite prijave. Sistem administratortima pruža mogućnost pregleda i obrade prijava, mijenjanja statusa, dodavanja napomena i exportovanja podataka. U radu je prikazan cijeli proces razvoja aplikacije, od analize zahtjeva i modeliranja sistema, preko ER dijagrama baze podataka i definisanja slučajeva korištenja, do implementacije pomoću Laravel frameworka i MySQL baze. Posebna pažnja posvećena je primjeni MVC arhitekture, objektno-relacionog mapiranja (ORM) i principima relacijskog modeliranja podataka, kao i integraciji OCR tehnologije za automatsku ekstrakciju ocjena iz dokumenata. Cilj rada je prikazati praktičnu primjenu savremenih web tehnologija u razvoju funkcionalne i dobro organizirane web aplikacije. Kroz implementaciju i testiranje sistema analizirana je ispravnost rada baze podataka, međusobna komunikacija komponenti i pouzdanost ključnih funkcija. Na taj način rad daje cjelovit prikaz procesa razvoja web aplikacije — od ideje i modeliranja do realizacije i provjere rada.

# 2 Modeliranje aplikacije

Modeliranje aplikacije obuhvata izradu dijagrama korištenja i ponašanja sustava prije implementacije. U nastavku je opisan predmetni sustav i prikazani su dijagrami na kojima se temelji projektovanje. [3].

## 2.1 Opis aplikacije

Cilj aplikacije je olakšati prijavu i obradu upisa studenata na Univerzitet u Bihaću korištenjem objektno orijentirane baze podataka i web tehnologija. Glavne klase su: User (korisnik), Application (prijava), Faculty (fakultet), Department (odsjek), Subject (predmet), ApplicationGrade (ocjena u prijavi), ScoringRule (pravilo bodovanja), Document (dokument) i AdminNote (administrativna napomena). U programu se rade objekti ovih klasa, a nad njima se izvršavaju odgovarajuće operacije, čime se prikazuje rad sa podacima u obliku objekata. Korištena je Laravel tehnologija, objektno orijentirani PHP framework za web aplikacije. Korištena je SQL baza podataka čiji su entiteti mapirani u objekte pri pokretanju Laravel aplikacije. Glavni sadržaj aplikacije čine prijava za upis, unos ocjena, automatski izračun bodova prema pravilima po odsjeku te upload i pregled dokumenata. Aplikacija sadrži navigacioni meni sa linkovima prema stranicama za kreiranje i uređivanje prijave, pregled fakulteta i odsjeka te administratorskom panelu za obradu prijava.

## 2.2 Statički UML dijagrami

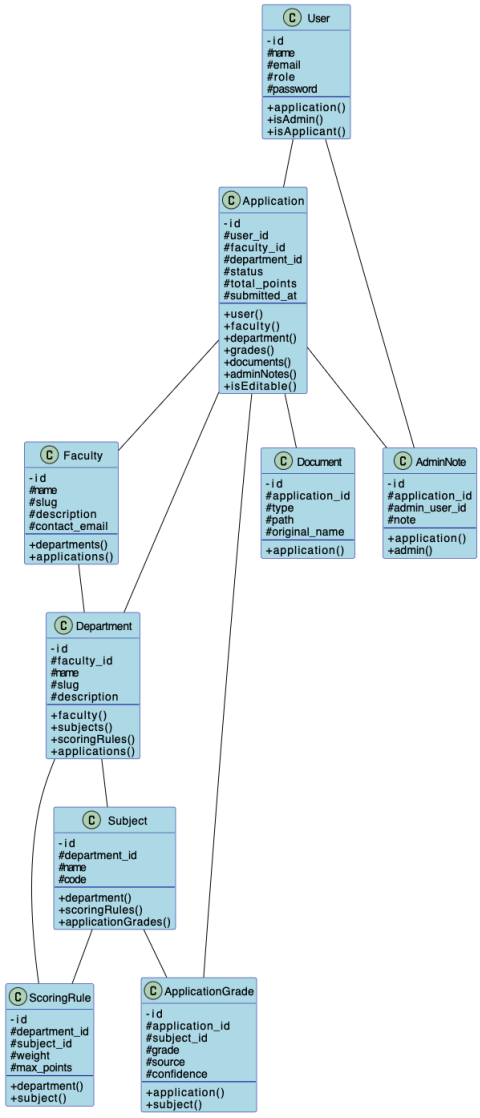
Statički UML (Unified Modelling Language) dijagrami služe za opis strukture sustava i njegovih osnovnih dijelova bez obzira na ponašanje u vremenu. Postoji više vrsta statičkih dijagrama:

- Dijagram strukture aplikacije
- Klasni diajgram
- Objektni dijagram
- Dijagram međusobnih odnosa objekata (kompozicioni dijagram)
- Dijagram pravila (ograničenja svakog dijela sistema)

Od navedenih statičkih dijagrama u nastavku je detaljnije opisan i prikazan klasni dijagram.

### 2.2.1 Klasni dijagram

Dijagram klasa opisuje klase koje su dio softverskog sistema. Termin klasa se koristi u objektno orijentiranom programiranju za definiranje strukture podataka koja se sastoji od svojstava (atributa) i funkcija (takođe se nazivaju metode). Klasni dijagram rađenog sistema je prikazan na slici 1.




Slika 1: Klasni dijagram

## 2.3 Dinamički UML dijagrami

Za modeliranje aplikacija, pored statičkih dijagrama, koriste se i dinamički dijagrami. Dinamički UML dijagrami opisuju ponašanje sistema, odnosno oni opisuju operacije, radnje i promjene koje se događaju u sistemu tokom rada. Postoji 5 vrsta dinamičkih dijagrama:

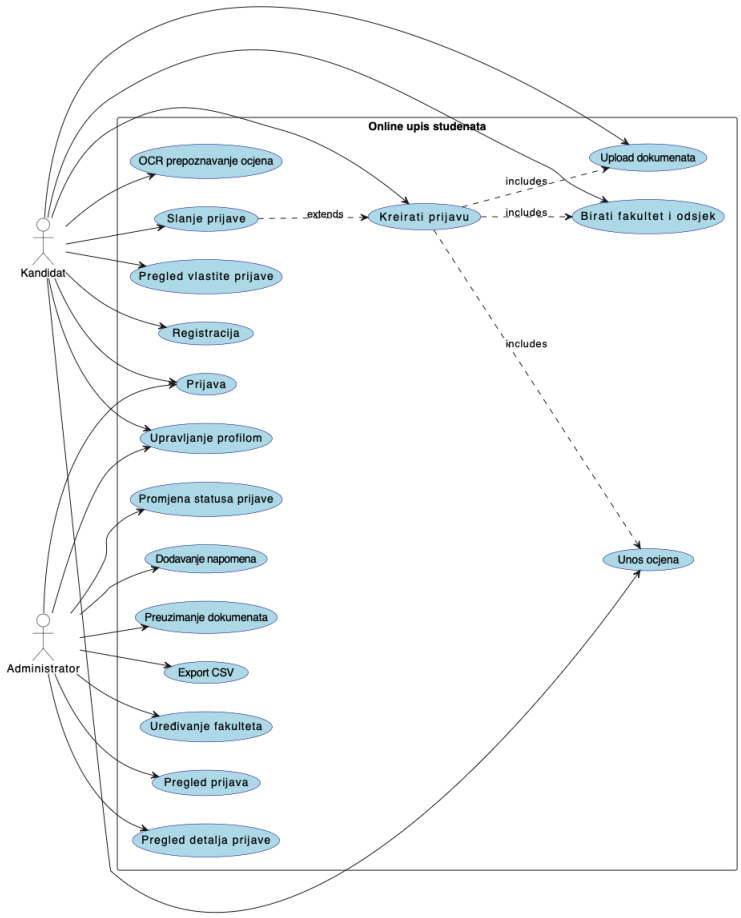
- Dijagram aktivnosti
- Dijagram stanja
- Dijagram slučajeva korištenja
- Sekvencijalni dijagram
- Komunikacioni dijagram

Od navedenih dinamičkih dijagrama u nastavku je detaljnije opisan i prikazan dijagram slučajeva korištenja.

### 2.3.1 Dijagram slučajeva korištenja

Dijagram slučajeva korištenja (engl. Use Case Diagram) je dinamički UML dijagram koji pokazuje funkcionalnosti sistema iz perspektive korisnika. Na njemu su prikazani akteri (korisnici, vanjski sistemi), slučajevi korištenja i veze između njih. U aplikaciji za online upis studenata postoje dva glavna aktera: Kandidat (applicant) – korisnik koji se prijavljuje za upis. Nakon registracije i prijave može kreirati prijavu, birati fakultet i odsjek, unositi ili ispravljati ocjene iz relevantnih predmeta, uploadovati dokumete (svjedodžbu, ličnu kartu, rodni list, dokaz uplate), koristiti OCR za automatsko prepoznavanje ocjena iz dokumenata, pregledati izračun bodova i poslati prijavu. Mogućnosti su ograničene na njegovu vlastitu prijavu i profil. Administrator (admin) – korisnik koji obrađuje prijave. Može pregledavati listu svih prijava sa filterima (po fakultetu, odsjeku, statusu, datumu), otvoriti detalje prijave (bodove, ocjene, dokumente), mijenjati status prijave (Draft, Submitted, Under review, Accepted, Rejected, Needs correction), dodavati napomene, preuzimati dokumente i exportovati podatke u CSV formatu. Administrator može i uređivati podatke o fakultetima. Oba aktera koriste zajedničke funkcionalnosti: registraciju, prijavu i upravljanje profilom (koje dolaze sa Laravel Jetstream paketom). Ovakva podjela omogućava jasnu kontrolu pristupa i sigurnost podataka. Dijagram slučajeva korištenja jasno prikazuje šta svaki akter može raditi u sistemu i kako sistem reaguje na njegove akcije.

Opisani dijagram slučajeva korištenja (Use-case dijagram) rađenog sistema je prikazan na slici 2.

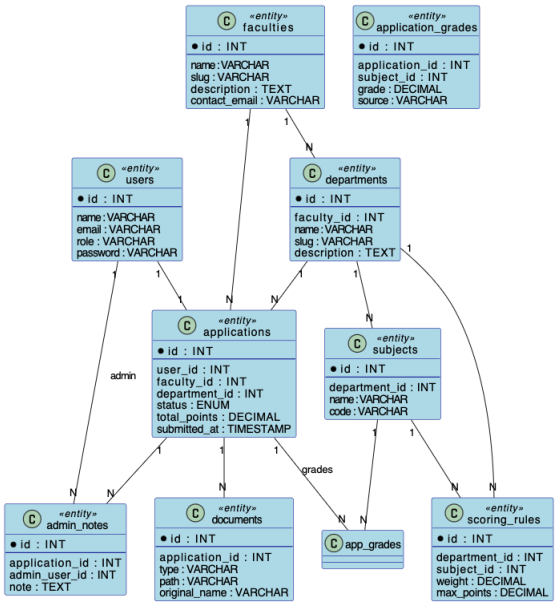



Slika 2: Use-Case dijagram

## 2.4 ER dijagram baze podataka

Baza podataka aplikacije za online upis studenata modelirana je pomoću ER (Entity-Relationship) dijagrama, koji prikazuje entitete, njihove atribute i veze između njih. U bazi postoji devet medusobno povezanih tabela. Tabela `users` čuva podatke o registrovanim korisnicima: ime, email, ulogu i lozinku. Svaki korisnik može imati najviše jednu prijavu, pa je veza između `users` i `applications` tipa 1:1. Tabela `applications` predstavlja prijave za upis i sadrži reference na korisnika, fakultet i odsjek, kao i status prijave, ukupne bodove, datum slanja i status ekstrakcije ocjena. Jedna prijava može imati više ocjena, više dokumenata i više administrativnih napomena. Tabela `faculties` sadrži fakultete sa nazivom, slug-om, opisom i kontakt podacima. Jedan fakultet ima više odsjeka i više prijava. Tabela `departments` predstavlja odsjeke povezane sa fakultetom, predmetima i pravilima bodovanja. Tabela `subjects` sadrži predmete po odsjecima. Tabela `application` povezuje prijave i predmete i čuva ocjene, izvor i nivo pouzdanosti. Tabela `scoring_rules` definiše za svaki odsjek i predmet težinu i maksimalne bodove. Tabela `documents` čuva putanje i metapodatke upload-ovanih dokumenata po prijavi. Tabela `admin_notes` sadrži napomene administratora po prijavi i upućuje na korisnika koji ju je dodao. Veze su uglavnom tipa 1:N. Primarni i strani ključevi osiguravaju referencijalni integritet i konzistentnost baze.

Opisani ER dijagram rađenog sistema je prikazan na slici 3.




Slika 3: Entity Relationship dijagram

# 3 Implementacija

## 3.1 Tehnologija izrade aplikacije

Aplikacija je implementirana u programskom jeziku PHP uz Laravel framework, Jetstream paket i Livewire komponente. Laravel je zasnovan na MVC (Model View Controller) arhitekturi, pa su poslovna logika, prikaz i upravljanje podacima jasno razdvojeni. Za bazu podataka korišten je MySQL.

### 3.1.1 Laravel

Laravel je besplatni, open-source PHP framework namijenjen razvoju web aplikacija u MVC arhitekturi. U projektu se koriste migracije za definisanje sheme baze, Eloquent ORM za rad sa modelima, Blade templati za prikaz i Livewire za reaktivne komponente (npr. višekoračni wizard prijave i upload dokumenata). Autentifikacija i uloge (kandidat, administrator) realizovani su preko Jetstream paketa. Laravel pruža routing, validaciju, sesije i rad sa fajlovima (Storage), što olakšava razvoj i održavanje aplikacije. [4].

### 3.1.2 SQL

MySQL je open-source sistem za upravljanje relacijskim bazama podataka. Podaci su organizirani u tabele sa primarnim i stranim ključevima. Shema baze definisana je Laravel migracijama, a pristup podacima ide preko Eloquent ORM-a, tako da se direktno pisanje SQL upita svede na minimum. SQL i dalje služi kao osnova za kreiranje tabela, definisanje veza, indeksa i ograničenja integriteta. [2].

## 3.2 MVC arhitektura

MVC (Model View Controller) je arhitekturni obrazac koji odvaja aplikaciju na tri sloja. Model upravlja podacima i poslovnom logikom (npr. `Application`, `User`, `Faculty`, `Department`). Kontroleri primaju HTTP zahtjeve, obrađuju ih i komuniciraju sa modelima i servisima (npr. `ApplicationController`, `ApplicationManagementController`). View (Blade šabloni i Livewire komponente) odgovoran je za prikaz podataka korisniku. Ovakvo razdvajanje olakšava testiranje, razvoj i održavanje aplikacije [1].

## 3.3 Objektno-relaciono mapiranje

Objektno-relaciono mapiranje (ORM, engl. Object-Relational Mapping) je tehnika koja omogućava rad sa relacijskom bazom podataka preko objekata i klasa u objektno-orijentiranom programskom jeziku. Umjesto ručnog pisanja SQL upita, operacije nad bazom (čitanje, dodavanje, ažuriranje, brisanje) izvode se pozivom metoda nad objektima, pri čemu ORM sloj prevodi te pozive u odgovarajuće SQL naredbe. Tako se postiže sloj apstrakcije između aplikacije i baze, što olakšava razvoj, smanjuje greške i omogućava lakše prebacivanje na drugi

DBMS ako je potrebno. U aplikaciji za online upis studenata korišten je Eloquent ORM koji dolazi sa Laravel frameworkom. Svaka tabela u bazi mapirana je na jednu PHP klasu (model).

```
// Rucni pristup bez ORM-a
$application_list = [];
$sql = "SELECT * FROM applications WHERE status = 'Submitted'
        AND faculty_id = ?";
$data = DB::select($sql, [$facultyId]);
foreach ($data as $row) {
    $application = new Application();
    $application->id = $row->id;
    $application->user_id = $row->user_id;
    $application->faculty_id = $row->faculty_id;
    $application->status = $row->status;
    $application->total_points = $row->total_points;
    $application_list[] = $application;
}
// Pristup koristeci ORM
$application_list = Application::where('status', 'Submitted')
    ->where('faculty_id', $facultyId)
    ->get();
```

Kod 1: Primjer korištenja ORM-a

Kao što se može vidjeti iz priloženo primjera, koristeći ORM veliki dio koda je unaprijed napisan i rad je znatno pojednostavljen. Jedino što je potrebno jeste pozvati neku od gotovih funkcija nad odgovarajućim objektom (tableom u bazi) i dobijaju se svi podaci kao i pisanjem potpunog upita, i programske logike koja razvrstava i dodijeljuje rezultate.

# 4 Analiza rada aplikacije

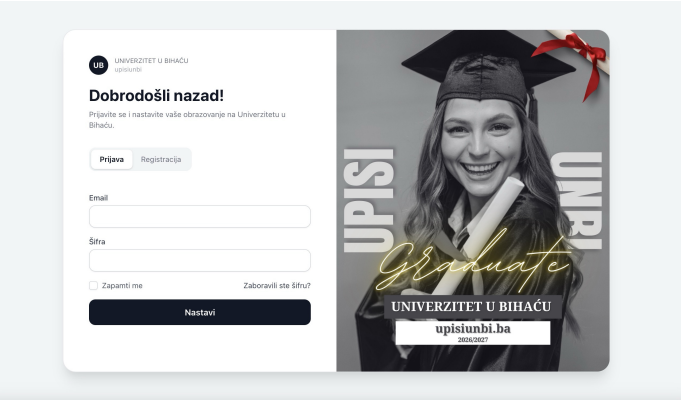
Analiza rada web aplikacije je izvršena prikazom pojedinačnih slučajeva korištenja. Prikazani su akteri određenog slučaja korištenja, tok procesa, te povratna informacija ili akcija koja može biti pozitivna ili negativna.

## 4.1 Opis slučajeva korištenja

**Slučaj korištenja 1: Prijava na sistem**

- Naziv SK: Prijava na sistem,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Postoji registrovan korisnik na sistem sa datim korisničkim računom,
- Osnovni scenarij SK:
  - Akter unosi email adresu i lozinku,
  - Sistem provjerava postoji li korisnik i ispravnost lozinke,
  - Ukoliko podaci tačni sistem otvara početnu stranicu (dashboard).
- Sporedni scenarij SK:
  - Ukoliko uneseni podaci nisu ispravni, sistem prikazuje poruku o grešci i omogućava ponovni unos.

Prikaz stranice za prijavu na sistem je predstavljen na slici 4.

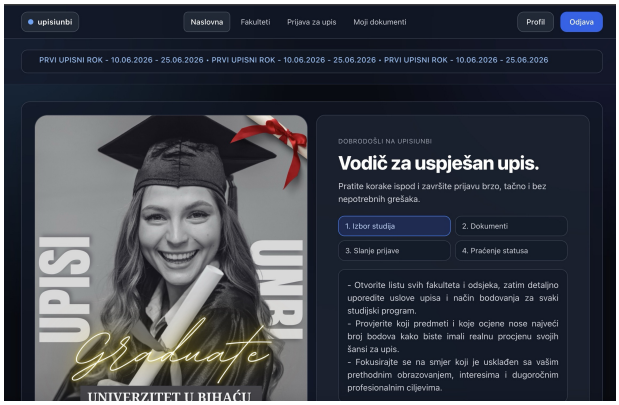



Slika 4: Stranica za prijavu

- Sporedni scenarij SK:

  - Unešeni podaci nisu registrovani u sistemu, sistem ispisuje poruku o netačnim podacima.

Prikaz početne stranice, u slučaju uspješne prijave na sistem je predstavljen na slici 5.




Slika 5: Početna stranica

**Slučaj korištenja 2: Registracija korisnika**

- Naziv SK: Registracija novog korisnika,
- Akteri SK: Neregistrovani korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Korisnik nema postojeći nalog u sistemu,
- Osnovni scenarij SK:

  - Akter otvara stranicu za registraciju,

  - Akter unosi ime, email adresu i lozinku,

  - Sistem provjerava validnost unesenih podataka,

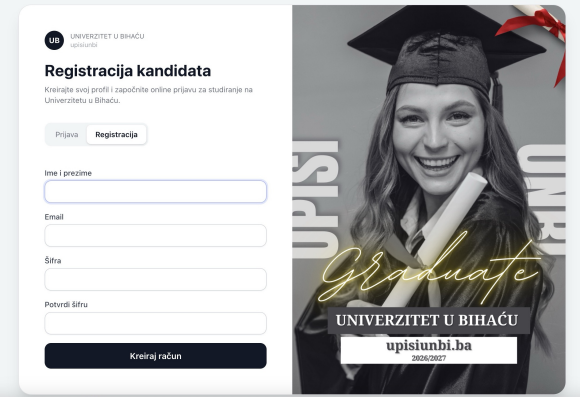
  - Ukoliko su podaci ispravni, sistem kreira novi korisnički nalog,

  - Korisnik može se prijaviti na sistem.

- Sporedni scenarij SK:

  - Ukoliko je email adresa već registrovan ili podaci nisu validni, sistem prikazuje odgovarajuću poruku o grešci.

Prikaz stranice za registraciju korisnika je predstavljen na slici 6.



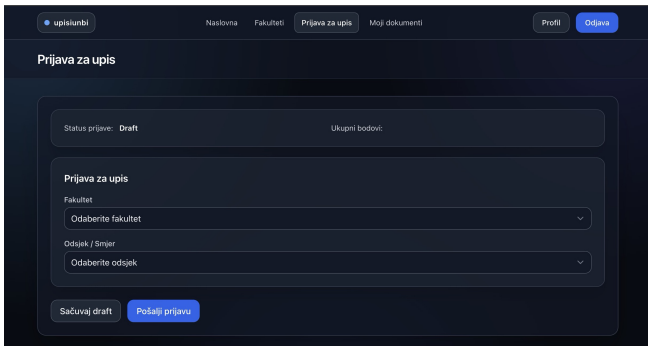

Slika 6: Stranica za registraciju korisnika

- Sporedni scenarij SK:
  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

**Slučaj korištenja 3: Kreiranje i uređivanje prijave**

- Naziv SK: Kreiranje ili uređivanje prijave za upis,
- Akteri SK: Kandidat (applicant),
- Učesnici SK: Akter i sistem,
- Preduslov: Korisnik je prijavljen sa ulogom kandidata,
- Osnovni scenarij SK:
  - Akter otvara stranicu za prijavu,
  - Akter bira fakultet i odsjek,
  - Sistem prikazuje predmete relevantne za odabrani odsjek,
  - Akter unosi ocjene ili koristi OCR za prijedlog ocjena iz dokumenata,
  - Akter sprema prijavu kao draft ili šalje prijavu.
- Sporedni scenarij SK:
  - Ukoliko je prijava već poslata, uređivanje nije moguće.
  - Ukoliko validacija ne prođe, sistem prikazuje poruke o grešci.

Prikaz stranice za kreiranje i uređivanje prijave je predstavljen na slici 7.




Slika 7: Stranica za kreiranje i uređivanje prijave

- Sporedni scenarij SK:

  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

**Slučaj korištenja 4: Upload dokumenata**

- Naziv SK: Upload dokumenata uz prijavu,
- Akteri SK: Kandidat,
- Učesnici SK: Akter i sistem,
- Preduslov: Korisnik je prijavljen i prijava je u statusu Draft ili Needs correction,
- Osnovni scenarij SK:

  - Akter odabire tip dokumenta (lična karta, svjedodžba, rodni list, dokaz uplate),

  - Akter uploaduje fajl,

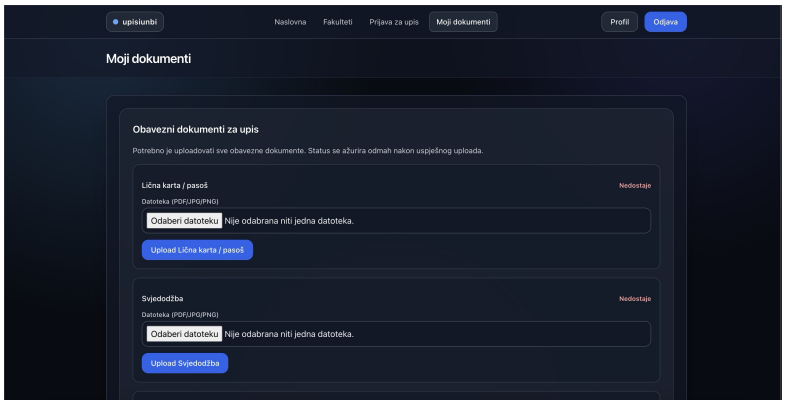
  - Sistem provjerava tip i veličinu fajla,

  - Sistem pohranjuje dokument u storage i povezuje ga sa prijavom.

- Sporedni scenarij SK:

  - Ukoliko fajl nije podržan ili je prevelik, sistem prikazuje poruku o grešci.

Prikaz stranice za upload dokumenataje predstavljen na slici 8.



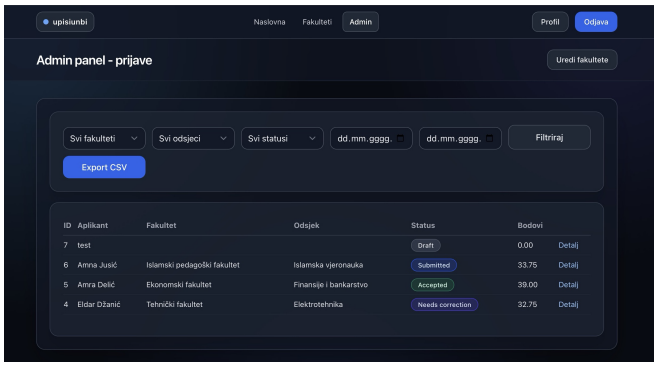

Slika 8: Stranica za upload dokumenata

- Sporedni scenarij SK:
  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

**Slučaj korištenja 5: Pregled prijava (administrator)**

- Naziv SK: Pregled liste prijava,
- Akteri SK: Administrator,
- Učesnici SK: Akter i sistem,
- Preduslov: Korisnik je prijavljen sa ulogom administratora,
- Osnovni scenarij SK:
  - Akter otvara stranicu admin panela za prijave,
  - Sistem prikazuje listu prijava sa osnovnim podacima,
  - Akter može filtrirati prijave po fakultetu, odsjeku, statusu ili datumu,
  - Akter odabire prijavu i otvara detalje.
- Sporedni scenarij SK:
  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Prikaz stranice za pregled prijave predstavljen na slici 8.




Slika 9: Stranica za pregled prijave

- Sporedni scenarij SK:
  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

**Slučaj korištenja 6: Export podataka u CSV**

- Naziv SK: Export prijava u CSV format,
- Akteri SK: Administrator,
- Učesnici SK: Akter i sistem,
- Preduslov: Administrator je prijavljen,
- Osnovni scenarij SK:
  - Akter odabire opciju export CSV,
  - Sistem generiše CSV fajl sa prijavama i povezanim podacima,
  - Sistem omogućava preuzimanje generisanog fajla.
- Sporedni scenarij SK:
  - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

## 4.2 Testiranje API-a

U ovom dijelu rada opisan je način testiranja API-ja aplikacije za online upis studenata. Za testiranje je korišten alat Insomnia, koji omogućava slanje HTTP zahtjeva prema serveru i pregled odgovora u JSON formatu. Posebna pažnja posvećena je provjeri HTTP status kodova i ispravnosti vraćenih podataka. Testirana je GET ruta `http://127.0.0.1:8000/api/statistics`, koja služi za dohvat osnovnih statistika aplikacije (broj fakulteta, odsjeka, predmeta i prijava). Slanjem zahtjeva putem Insomnia server vraća odgovor u JSON formatu sa brojem fakulteta, odsjeka, predmeta i registrovanih kandidata. U slučaju uspješnog zahtjeva server vraća status kod 200 OK i strukturirane JSON podatke. Time je potvrđeno da API ispravno dohvaća podatke iz baze putem Eloquent ORM-a i da pravilno obrađuje zahtjeve klijenta. Na slici 15 prikazan je primjer testiranja rute `/api/statistics` u alatu Insomnia. Testiranje je izvršeno u razvojnom okruženju kako bi se provjerila funkcionalnost API-ja prije produkcijske upotrebe.

# 5 Zaključak

U ovom radu prikazan je razvoj i implementacija web aplikacije za online prijavu i obradu upisa studenata na Univerzitet u Bihaću. Proces je obuhvatio analizu zahtjeva, modeliranje sistema, projektovanje baze podataka i implementaciju primjenom Laravel frameworka i MySQL baze. Modeliranjem su definisani glavni akteri sistema (kandidat i administrator), slučajevi korištenja, kao i struktura baze putem ER dijagrama i klasnog dijagrama. Posebna pažnja posvećena je referencijalnom integritetu i pravilnim vezama između tabela, što je omogućilo stabilan rad sistema. Implementacija je izvršena uz Laravel, Jetstream i Livewire, primjenom MVC arhitekture i Eloquent ORM-a. Odvajanje modela, kontrolera i prikaza doprinijelo je preglednosti i modularnosti koda, a ORM je pojednostavio rad sa bazom putem objekata umjesto direktnog pisanja SQL upita. MySQL baza koristi migracije, primarne i strane ključeve te Laravel Storage za sigurno čuvanje dokumenata. Implementirane su funkcije registracije i prijave, kreiranja i uređivanja prijave, unosa ocjena, automatskog izračunavanja bodova, upload dokumenata i OCR ekstrakcije ocjena iz PDF-a i skenova, kao i administratorski panel za pregled prijava, promjenu statusa, dodavanje napomena i export u CSV. Testiranjem je provjerena ispravnost rada baze, OCR integracije i tokova prijave. Na osnovu izvršenog testiranja može se zaključiti da aplikacija ispravno funkcioniše i da je ostvarena pravilna komunikacija između korisničkog interfejsa, poslovne logike i baze podataka. Strukturiran pristup razvoju uz primjenu savremenih tehnologija rezultirao je funkcionalnom i organizovanom web aplikacijom za online upis studenata. Dalji razvoj može uključivati e-mail obavještenja o promjeni statusa prijave, naprednije filtere i pretragu u admin panelu, PDF izvještaje te eventualnu mobilnu verziju aplikacije.

# Literatura


[1] Object Management Group (OMG), *Unified Modeling Language Specification*, Version 2.5.1, https://www.omg.org/spec/UML/, pristup: februar 2026.

[2] MDN Web Docs, MVC (Model-View-Controller), https://developer.mozilla.org/en-US/docs/Glossary/MVC, pristup: februar 2026.

[3] MySQL 8.0 Reference Manual, Oracle Corporation, pristup: februar 2026. .

[4] Laravel dokumentacija, Dostupno na: https://laravel.com/docs

[5] Laravel Jetstream dokumentacija, Dostupno na: https://jetstream.laravel.com