Razvoj web aplikacije za upis u višu godinu studija

Fabac, Ana

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni diplomski studij Informatika

Ana Fabac

Razvoj web aplikacije za upis u višu godinu studija

Diplomski rad

Mentor: doc. dr. sc. Martina Ašenbrener Katić

Rijeka, veljača 2024

Sveučilište u Rijeci
FIDIT Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 12. lipnja 2023.

Zadatak za diplomski rad

Pristupnik: Ana Fabac

Naziv diplomskog rada: Razvoj web aplikacije za upis u višu godinu studija

Naziv diplomskog rada na eng. jeziku: Development of a web application for enrollment in higher years of study

Sadržaj zadatka:

Zadatak diplomskog rada je opisati postupak planiranja i stvaranja web aplikacije za upis u višu godinu studija. Aplikacija će obuhvaćati evidenciju podataka o studentima, predmetima te zadovoljenim uvjetima za upis u višu godinu. Osnovna funkcionalnost aplikacije je upis studenta u višu godinu studija te odabir izbornih predmeta. Procesu izrade aplikacije prethodit će analiza poslovnog sustava te na temelju toga oblikovanje modela podataka. Aplikacija će biti izrađena u Laravelu te je potrebno opisati postupak izrade aplikacije.

Mentor:
Doc. dr. sc. Martina Ašenbrener Katić
Martina Ašenbrener Katić

Voditeljica za diplomske radove:
Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 15. lipnja 2023

(Ana Fabac)
Ana Fabac

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: **+385(0)51 584 700**
E-mail: **ured@inf.uniri.hr**
OIB: **64218323816**
IBAN: **HR1524020061400006966**

SAŽETAK

U diplomskom radu opisan je postupak izrade aplikacije za upis u više godine studija. Aplikacija je izrađena u Laravel 10 web okviru zasnovanom na PHP-u. Prije same izrade aplikacije opisan je postupak upisa u više godine te po opisanom procesu upisa izrađen je model podataka na kojoj se temelji izrada web aplikacije. Osnovna funkcionalnost aplikacije je upis u višu godinu studija kod kojega se upisuju odgovarajući podaci za upise te odabiru izborni predmeti. U radu detaljno je opisan model podataka i veze između svih tablica, nakon čega slijedi detaljan opis izrade aplikacije i prikaz tijeka korištenja aplikacije iz perspektive admina i studenta.

Ključne riječi: Laravel, upis u više godine studija, tablica, student, upisi, admin

SADRŽAJ

1. UVOD

Zadatak ovog diplomskog rada je izrada aplikacije za upise u višu godinu studija u Laravelu i opisati postupak izrade aplikacije. Laravel je PHP[1] okvir otvorenog koda koji sadrži sve potrebne komponente i značajke za izradu web stranice [1]. Kreator Laravela Taylor Otwell je objavio prvu verziju Laravela u lipnju 2011. godine kako bi riješio nedostatke fleksibilnosti i jednostavnosti PHP okvira CodeIgniter [2]. Druga verzija Laravela je objavljenja iste godine u studenom kada se u Laravel uvodi Model-Pogled-Upravitelj (eng. Model-View-Controller) ili MVC arhitektura s kojom se donosi red i dosljednost u nestrukturiranom kodu čime se olakšava i sam razvoj i održavanje web aplikacija [3]. MVC arhitektura funkcionira na način da model predstavlja strukturu tablice u bazi podataka, dok upravitelj komunicira s modelom i dobiva podatke od modele nakon čega izvršava funkcije nad podacima te ih šalje pogledu u svrhu prikazivanja korisniku.

Proces upisa u višu godinu studija naporan je i stresan za mnoge studente zbog količine administrativnog posla te mogućnosti same pogreške prilikom ispunjavanja dokumenata i predaje svih potrebnih i dodatnih dokumenata u roku. U samom procesu upisa potrebno je ispuniti podatke o studentu koji se upisuje nakon čega se ispunjavaju podaci o studiju studenta te položenim kolegijima i upisanim godinama. Nakon toga, ispunjavaju se podaci o upisu u tekuću akademsku godinu te biraju se izborni predmeti za zimski i ljetni semestar. Prilikom tog procesa potrebno je svaki obrazac pronaći na mrežnim stranicama fakulteta, isprintati, ispuniti te poslati mailom ili osobno donijeti u studensku službu. Ovakav način upisa je nepraktičan i kaotičan za studentsku službu ukoliko postoji veliki broj studenata te predstavlja veliku razinu stresa studentima zbog brige o nedostatku ili krivo ispunjenoj dokumentaciji. Izradom aplikacije u Laravalu želi se olakšati sam proces upisa na način da se automatizira što više procesa kako bi se smanjila ljudska pogreška te sve potrebne informacije bi se nalazile na jednom mjestu.

Diplomski rad je podijeljen u pet glavnih poglavlja. U prvom poglavlju se opisuju osnovni koncepti modela podataka te je prikazan model podataka za upise u više godine studija popraćen detaljnim opisom svih entiteta modela kao i njihovih veza. U drugom poglavlju započinje izrada aplikacije s opisom pripreme radnog okruženja i izrada i uređivanje tablica. U trećem poglavlju opisuje se postavljanje ograničenja pristupa podacima ovisno u ulozi prijavljenog korisnika i opisuje se ograničavanje prikaza izbornika na navigaciji. Dok u četvrtom poglavlju se opisuje postupak kreiranja različitog prikaza iste tablice i ograničavanje akcija nad podacima. U posljednjom poglavlju se prolazi tijek aplikacije kroz nekoliko tablica za studenta i admina.

[1] PHP (akronim za Hypertext Preprocessor) je skripni jezik otvorenog koda za izradu web aplikacija ili stranica [4].

2. MODEL PODATAKA

Prije svake izrade aplikacije potrebno je definirati model podataka pomoću kojeg se definiraju veze između entiteta i njihovi atribute te u konačnici preko modela se prikazuje i sama poslovna organizacija. Osnovni koncepti svakog modela podataka jesu entitet i tip entiteta, veza i tip veze, atribut i brojnost [5].

Razrađen model entiteta i veza za upise u više godine studija počinje s prvim tipom entiteta DRZAVA kojemu je primarni ključ *Id_drzave* te sadrži atribut *Naziv_drzave*. Tip entitet DRZAVA je povezan sa slabim tipom entiteta ZUPANIJA na način da je jedna država iz tipa entiteta DRZAVA povezana s nula ili više županija iz tipa entiteta ZUPANIJA. Dok jedna županija iz tipa entiteta ZUPANIJA pripada jednoj i samo jednoj državi iz tipa entiteta DRZAVA. Primarni ključ slabog tipa entiteta ZUPANIJA sastavljen je od primarnog ključa tipa entiteta DRZAVA (*Id_drzave)* i svog primarnog ključa (*Id_zupanije*) te slabi tip entiteta ZUPANIIJA sadrži i atribut *Naziv_zupanije.* Slabi tip entiteta ZUPANIJA je povezan sa slabim tipom entitetom OPCINA tako da je OPCINA slabi tip entiteta od slabog tipa entiteta ZUPANIJA. Čime se dobiva da je jedna županija iz slabog tipa entiteta ZUPANIJA povezana s niti jednom ili više općina iz slabog tipa entiteta OPCINA, dok jedna općina iz slabog tipa entiteta OPCINA pripada jednoj i samo jednoj županiji iz slabog tipa entiteta ZUPANIJA. Što znači da primarni ključ entiteta OPCINA se sastoji od primarnog ključa slabog tipa entiteta ZUPANIJA (*Id_zupanije_Id_drzave*) i primarnog ključa slabog tipa entiteta OPCINA *Postanski_br* te sadrži atribut *Naziv_opcine.* Također, slabi tip entiteta OPCINA je povezan sa slabim tipom entiteta MJESTO koji sadrži informacije o mjestu kroz atribut *Naziv_mjesta*. Tip entiteta MJESTO je slabi tip entitet od slabog tipa entiteta OPCINA pa primarni ključ slabog tipa entiteta se sastoji od primarnog ključa slabog tipa entiteta OPCINA *Postanski_br_Id_zupanije_Id_drzave* i svog primarnog ključa *Id_mjesta*. Veza sa slabim tipom entiteta OPCINA je specijalna veza uz slabi tip entiteta MJESTO što znači da jedna općina iz slabog tipa entiteta OPCINA povezana je s niti jednom ili mnogo mjesta iz slabog tipa entitet MJESTO. Dok uz jedno mjesto iz slabog tipa entiteta MJESTO pripada jednoj i samo jednoj općini iz slabog tipa entitet OPCINA.

Također, slabi tip entiteta MJESTO je povezan s slabim tipom entiteta FAKULTET i tipom entiteta STUDENT. Na način da jedan fakultet iz slabog tipa entiteta FAKULTET se nalazi u jednom i samom jednom mjesto iz slabog tipa entiteta MJESTO, dok jedno mjesto iz slabog tipa entiteta MJESTO je povezano s niti jednim ili mnogo fakulteta iz slabog tipa entiteta FAKULTET. Dok veza s tipom entiteta STUDENT je definirana tako da jedan student iz tipa entiteta STUDENT povezana je s jednim i samo jednim mjestom iz slabog tipa entiteta MJESTO za prebivalište i mjesto rođenja studenta. Mjesto iz slabog tipa entiteta MJESTO povezan je s niti jednim ili mnogo studenata iz tipa entiteta STUDENT. Za boravište studenta jedno mjesto iz tipa entiteta MJESTO je povezan s niti jednim ili mnogo studenata iz tipa entiteta STUDENT, dok student iz tipa entiteta STUDENT je povezan s niti jednim ili jednim mjestom iz slabog tipa entiteta STUDENT.

Tip entiteta SVEUCILISTE sadrži informacije o sveučilištu kroz atribut *Naziv_sveucilista* te primarnog ključa *Id_sveucilista*. Povezan je slabim tipom entiteta FAKULTET odnosno kako

je slabi tip entitet FAKULTET ovisan o tipu entitetu SVEUCILISTE uz tip entitet FAKULTET ne upisuje se brojnost pošto se radi o specijalnoj vezi čime slijedi da jedan fakultet iz slabog tipa entiteta FAKULTET pripada jednom i samo jednom sveučilištu iz tipa entiteta SVEUCILISTE. Dok jedno sveučilište iz tipa entiteta SVEUCILISTE je povezano s niti jednim ili mnogo fakulteta iz slabog tipa entiteta FAKULTET. Kako je tip entitet FAKULTET slabi tip entitet od tipa entiteta SVEUCILISTE njegov se primarni ključ sastoji od primarnog ključa tipa entiteta SVEUCILISTE *Id_sveucilista* i svog primarnog ključa *Id_fakulteta*. Tip entitet FAKULTET sadrži informacije o fakultetu kroz atribute *Naziv_fakulteta* i *Adresa_fakulteta* te povezan je i s tipom entitetom STUDENT. Jedan fakultet iz slabog tipa entiteta FAKULTET je povezan s jednim ili mnogo studenata iz tipa entiteta STUDENT, dok jedan student iz tipa entiteta STUDENT pohađa jedan i samo jedan fakultet iz slabog tipa entiteta FAKULTET.

Tip entiteta UZDRZAVATELJ je slabi tip entitet od tipa entiteta STUDENT te sadrži informacije o uzdržavateljima studenta preko atributa *Srodstvo* i primarnog ključa koji se sastoji od primarnog ključa tipa entiteta STUDENT *JMBAG* i svog primarnog ključa *Id_uzdrzavatelja*. Slabi tip entiteta UZDRZAVATELJ je povezan s tipovima entiteta STRUCNA_SPREMA, OBRAZOVANJE, ZANIMANJE, POLOZAJ_U_ZANIMANJU i STUDENT. Stručna sprema iz tipa entiteta STRUCNA_SPREMA povezana je s niti jednim ili mnogo uzdržavatelja iz slabog tipa entiteta UZDRZAVATELJ, dok jedan uzdržavatelj iz slabog tipa entiteta UZDRZAVATELJ ima jednu i samo jednu stručnu spremu iz tipa entiteta STRUCNA_SPREMA. Veza s tipom entiteta OBRAZOVANJE definirana je na način da jedno obrazovanje iz tipa entiteta OBRAZOVANJE povezana je s niti jednim ili mnogo uzdržavatelja iz slabog tipa entiteta UZDRZAVATELJ, dok jedan uzdržavatelj iz slabog tipa entiteta UZDRZAVATELJ ima jednu i samo jedno obrazovanje iz tipa entiteta OBRAZOVANJE. Veza s tipom entiteta ZANIMANJE je definirana na način jedno zanimanje iz tipa entiteta ZANIMANJE povezana je s niti jednim ili mnogo uzdržavatelja iz slabog tipa entiteta UZDRZAVATELJ te jedan uzdržavatelj iz slabog tipa entiteta UZDRZAVATELJ ima jednu i samo jedno zanimanje iz tipa entiteta ZANIMANJE. Vezu s tipom entiteta POLOZAJ_U_ZANIMANJU može se definirati tako da jedan položaj u zanimanju iz tipa entiteta POLOZAJ_U_ZANIMANJU je povezan s niti jednim ili mnogo uzdržavatelja iz slabog tipa entiteta UZDRZAVATELJ, dok jedan uzdržavatelj iz slabog tipa entiteta UZDRZAVATELJ ima jedan i samo jedan položaj u zanimanju iz tipa entiteta POLOZAJ_U_ZANIMANJU. Tip entiteta POLOZAJ_U_ZANIMANJU sadrži informacije o položajima u zanimanju te se tip entiteta sastoji od primarnog ključa *Id_polozaja_u_zanimanju* i atributa *Naziv_polozaja_u_zanimanju.* Tip entiteta ZANIMANJE sadrži informacije o zanimanju te se sastoji od primarnog ključa Id_*zanimanja* i atributa *Naziv_zanimanja*. Tip entiteta OBRAZOVANJE sadrži informacije o razinama obrazovanja kroz atribut *Naziv_obrazovanja* i primarnog ključa *Id_obrazovanja.* Te tip entiteta STRUCNA_SPREMA sadrži informacije o stručnoj spremi kroz atribute *Naziv_strucne_spreme, Kratica* i primarnog *ključa Id_strucne_sprema*.

Tip entiteta DRZAVLJANSTVO sadrži informacije o državljanstvu kroz atribut *Naziv_drzavaljanstva* i primarnog ključa *Id_drzavljanstva.* Tip entiteta povezan je s tipom entiteta STUDENT na način da jedno državljanstvo iz tipa entiteta DRZAVLJANSTVO

pripada nijednim ili mnogo studenta iz tipa entiteta STUDENT, dok jedan student iz tipa entiteta STUDENT ima jedno i samo jedno državljanstvo iz tipa entiteta DRZAVLJANSTVO. Također, tip entiteta NARODNOST sadrži informacije o narodnosti kroz atribut *Naziv_narodnosti* i primarnog ključa *Id_ narodnosti.* Tip entiteta je povezan s entitetom STUDENT tako da jedna narodnost iz tipa entiteta NARODNOST je povezana s niti jednim ili mnogo studenata iz tipa entiteta STUDENT, dok jedan student iz tipa entiteta STUDENT ima jednu i samo jednu narodnost iz tipa entiteta NARODNOST.

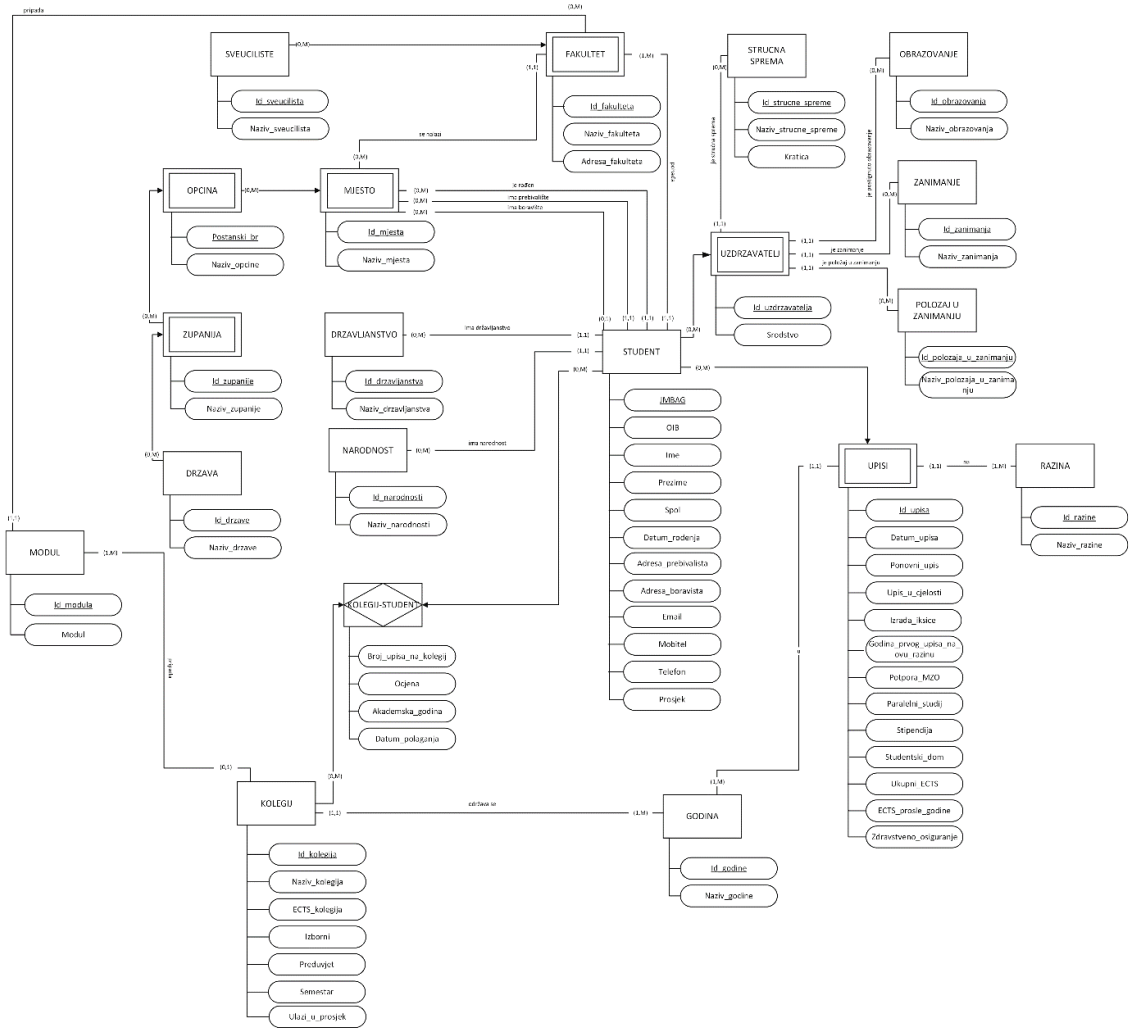
Tip entiteta STUDENT sadrži informacije o studentu kroz atribute *JMBAG*, koji je ujedno i primarni ključ, *OIB, Ime, Prezime, Spol, Datum_rodenja, Adresa_prebivalista, Adresa_boravista, Email, Mobitel, Telefon* i *Prosjek.* Osim prije navedenih veza, tip entiteta STUDENT povezan je i sa slabim tipom entitetom UPISI i agregiranim tipom entiteta KOLEGIJ_STUDENT. Jedan student iz tipa entiteta STUDENT je povezana s niti jednim ili mnogo upisima iz slabog tipa entiteta UPISI, dok jedan upis iz slabog tipa entiteta UPISI pripada jednom i samo jednom studentu iz tipa entiteta STUDENT. Tip entiteta UPISI je slab tip entiteta od tipa entiteta STUDENT te njegov primarni ključ se tvori od primarnog ključa tipa entiteta STUDENT (*JMBAG)* i svog primarnog ključa*(Id_upisa*). Tip entiteta UPISI se sastoji od atributa *Datum_upisa, Ponovni_upis, Upis_u_cjelosti, Izrada_iksice, Godina_prvog_upisa_na_ovo_razinu, Potpora_MZO, Paralelni_studij, Studentski_dom, Zdravstveno_osiguranje, Ukupni_ECTS, ECTS_prosle_godine, Stipendija.*

Slabi tip entiteta UPISI je povezan s tipom entitetima RAZINA i GODINA. Veza s tipom entiteta RAZINA definira se tako da jedna razina iz tipa entiteta RAZINA je povezana s jednim ili mnogo upisa iz slabog tipa entiteta UPISI, dok jedni upisi iz slabog tipa entiteta UPISI ima jednu i samo jednu razinu iz tipa entiteta RAZINA. Tip entiteta RAZINA sastoji se od atributa *Naziv_razine* i primarnog ključa *Id_razine* te sadrži informacije o razini. Veza tipa entiteta UPISI s tipom entitetom GODINA može se opisati na način da jedna godina iz tipa entiteta GODINA pripada jednim ili mnogo upisa iz slabog tipa entiteta UPISI, a jedni upisi iz slabog tipa entiteta UPISI upisuju se u jednu i samo jednu godinu iz tipa entiteta GODINA. Tip entiteta GODINA sastoji se od atributa *Naziv_godine* i primarnog ključa *Id_godine* te sadrži informacije o godini. Tip entiteta GODINA je povezan s tipom entitetom KOLEGIJ i to na način da jedna godina iz tipa entiteta GODINA je povezana s jednim ili mnogo kolegijima iz tipa entiteta KOLEGIJ, dok jedan kolegij iz tipa entiteta KOLEGIJ održava se na jednoj i samo jednoj godinu iz tipa entiteta GODINA. Tip entiteta KOLEGIJ sadrži informacije o kolegiju kroz atribute *Naziv_kolegija, ECTS_kolegija, Izborni, Preduvjet, Semestar, Ulazi_u_prosjek* te sadrži i primarni ključ *Id_kolegija*.

Tip entiteta KOLEGIJ je povezan s tipom entitetom MODUL i agregiranim tipom entiteta KOLEGIJ_STUDENT. Veza s tipom entiteta MODUL može se opisati tako da jedan kolegij iz tipa entiteta KOLEGIJ pripada nijednom ili jednom modulu iz tipa entiteta MODUL, dok jedan modul iz tipa entiteta MODUL povezan je s jednim ili mnogo kolegija iz tipa entiteta KOLEGIJ. Tip entiteta MODUL sastoji se od atributa *Naziv_modula* i primarnog ključa *Id_modula* te sadrži informacije o modulu. Tip entiteta MODUL je povezan i sa slabim tipom entiteta FAKULTET i to na način da je jedan modul iz tipa entiteta MODUL povezan s jednim i samo

jednim fakultetom iz slabog tipa entiteta FAKULTET, dok jedan fakultet iz slabog tipa entiteta FAKULTET povezan je s nijednim ili mnogo modula iz tipa entiteta MODUL.

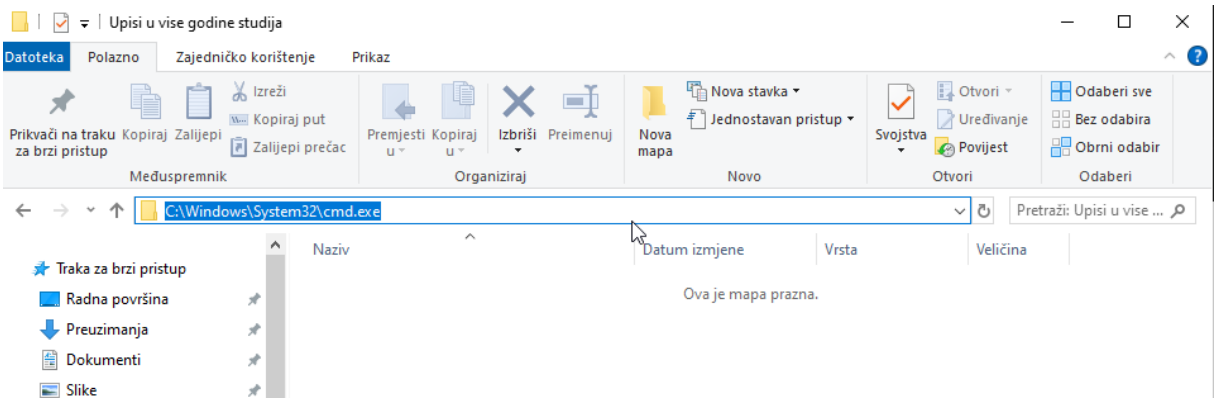
Agregirani tip entiteta KOLEGIJ_STUDENT sadrži informacije o kolegiju koje polaže student kroz atribute *Broj_upisa_na_kolegiju*, *Ocjena*, *Akademska_godina* i *Datum_polaganja*. Primarni ključ agregiranog tipa entiteta se tvori od primarnih ključeva njenih jakih tipova entiteta, u ovom slučaju za agregiran tip entiteta KOLEGIJ_STUDENT primarni ključ se tvori od primarnog ključa tipa entiteta KOLEGIJ (*Id_kolegija)* i primarnog ključa tipa entiteta STUDENT (*JMBAG)*. Pri čemu dolazi da jedan entitet kolegij_student iz agregiranog tipa entiteta KOLEGIJ_STUDENT pripada jednom i samo jednom studentu iz tipa entiteta STUDENT i jednom i samo jednom kolegiju iz tipa entiteta KOLEGIJ, dok student iz tipa entiteta STUDENT je povezan s niti jednim ili mnogo entiteta kolegij_student iz agregiranog tipa entiteta KOLEGIJ_STUDENT. Te isto vrijedi i za tip entitet kolegij, za kojeg jedan kolegij iz tipa entiteta KOLEGIJ je povezna povezan s niti jednim ili mnogo entiteta kolegij _student iz agregiranog tipa entiteta KOLEGIJ_STUDENT. Slika cijelog modela podataka je prikazana na Slici 1.



Slika 1. Prikaz modela podataka za upise u višu godinu studija

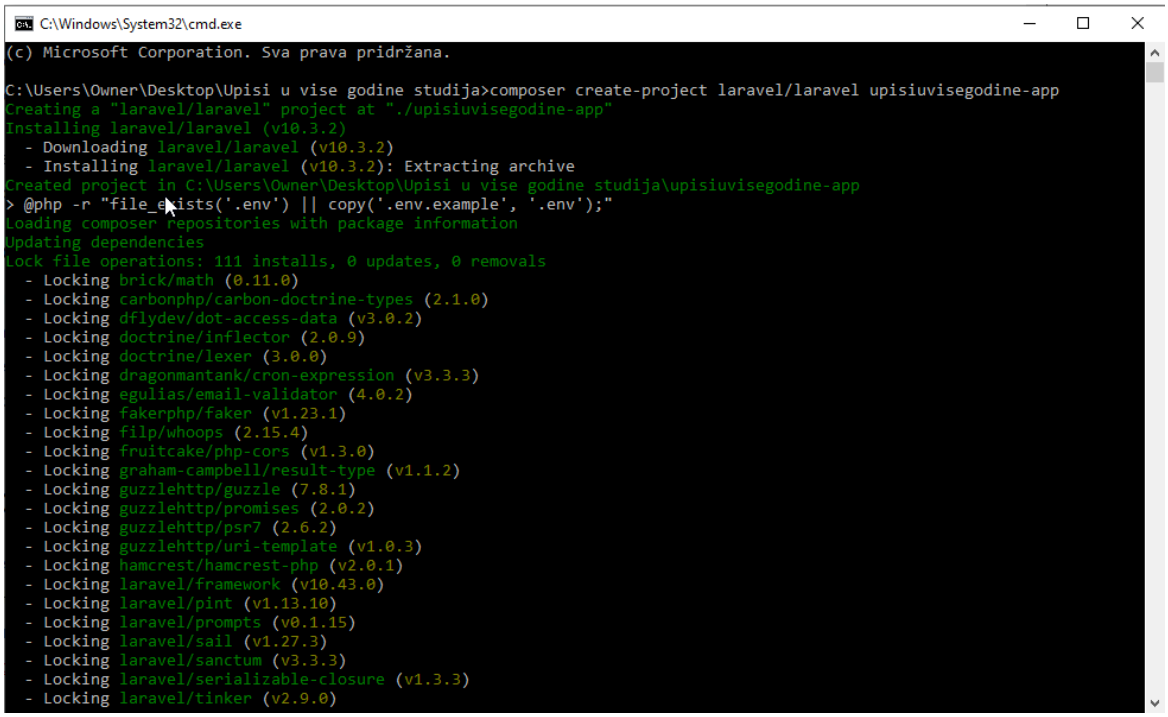
3. IZRADA APLIKACIJE U LARAVELU

Za izradu aplikacije upisi u više godine potrebno je imati instaliran PHP i Composer kako bi se mogao napraviti novi Laravel projekt [6]. Prvo je potrebno kreirati novu mapu *„Upisi u vise godine“* i otvoriti je, nakon čega kliknuti na putanju u mapi te upisati *cmd* pa kliknuti tipku enter kako bi se otvorio cmd terminal okvir u kojem se može kreirati projekt.



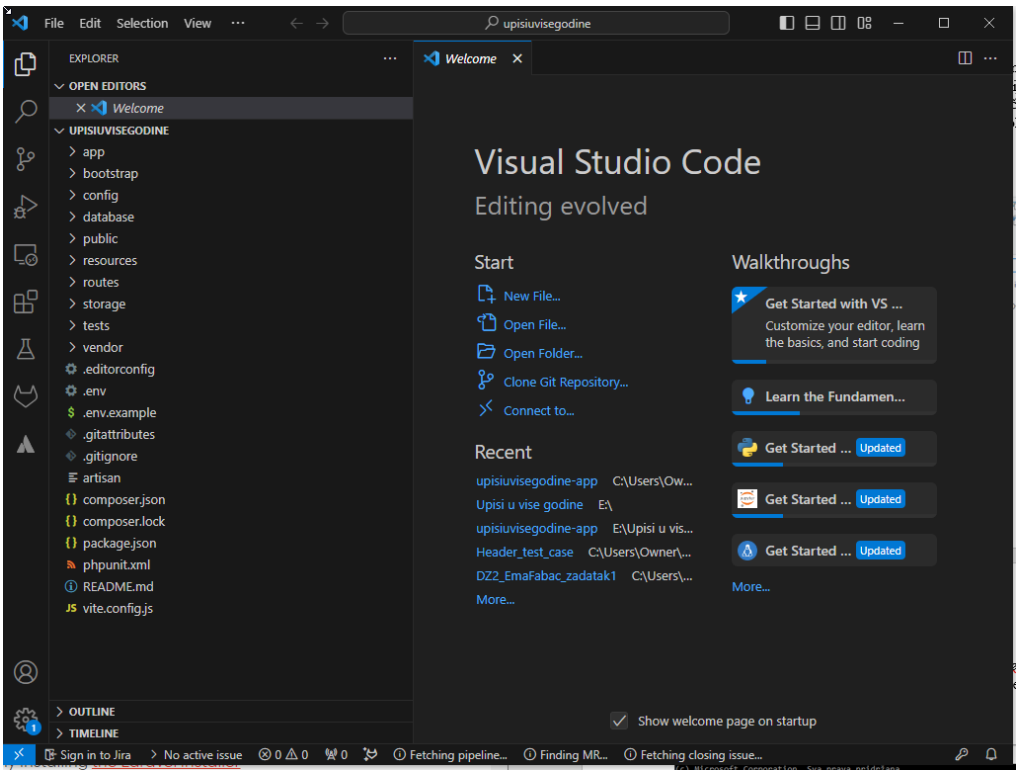
Slika 2. Prikaz kreiranje mape i otvaranje cmd terminala kroz putanju

U cmd terminalu potrebno je upisati naredbu *composer create-project laravel/laravel upisiuvisegodine-app* kako bi se instalirao laravel i kreirao novi laravel projekt u kojem su dodane sve datoteka koje postavljaju laravel okruženje.



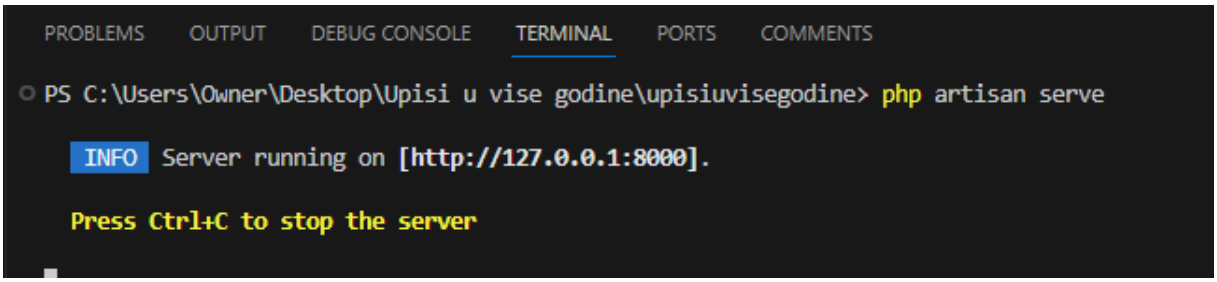
Slika 3. Prikaz kreiranja novog laravel projekta

Nakon uspješnog kreiranja projekta potrebno ga je otvoriti u *Visual Studio Code* aplikaciji preko koje se uređuje kod za Laravel projekt. Prikaz otvorenog projekta u Visual Studio uz sve datoteke koje su instalirane za postavljanje Laravel okruženja je prikazan na Slici 4.

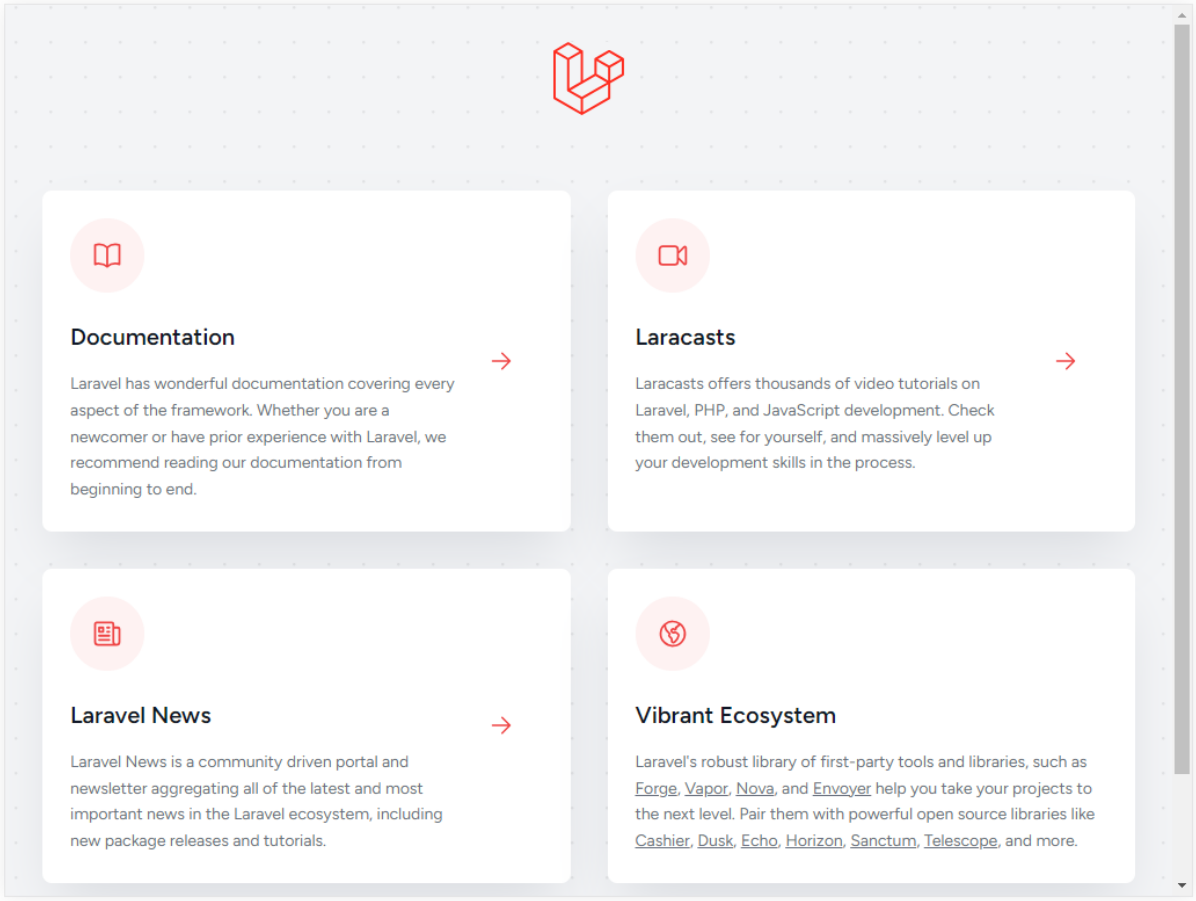


Slika 4. Prikaz projekta u Visual Studio Code

U aplikaciji je potrebno otvoriti terminal te upisati naredbu *php artisan serve* koja pokreće server za aplikaciju i u terminalu ispiše link koji otvara aplikaciju u web pregledniku. U pregledniku se prikazuje *welcome* stranica koja se definirana unaprijed te je kreirana prilikom kreiranja projekta. U aplikacije kod stranica se nalazi u mapi *resourse* pod *views* te datoteka ima nastavak *blade* koji je zapravo laravel nastavak i nastavak *php* pošto laravel se temelji na *php-u* čime se dolazi da cijeli naziv datoteke je *welcome.blade.php*.



Slika 5. Prikaz terminala nakon pokretanja naredbe php artisan serve



Slika 6. Prikaz welcome stranice

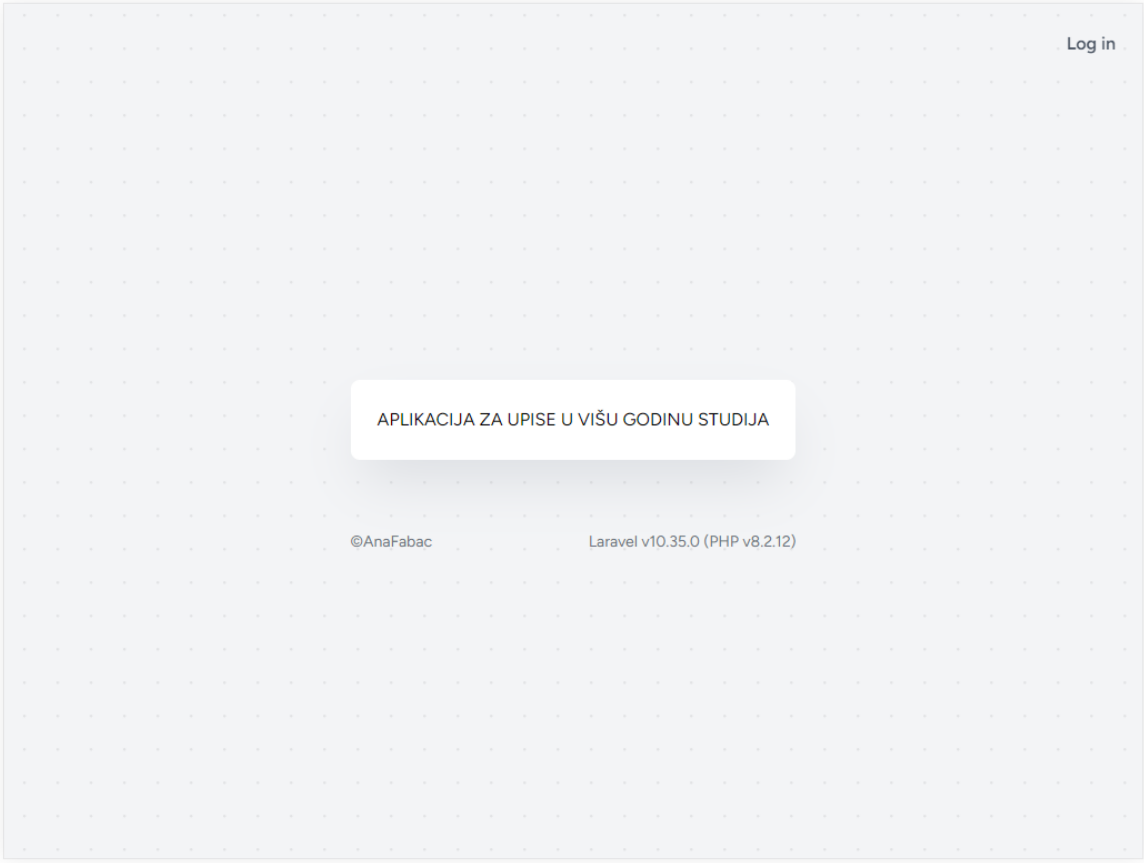
Datoteku *welcome.blade.php* je potrebno urediti kako bi se na njoj prikazo sadržaj koji je namijenjen za aplikaciju upisi u više godine studija. Sekciju *<head>* nije potrebno urediti, dok u sekciji *<body>* potrebno je izbrisati trenutni sadržaj i dodati novi * * u kojem se upisuje naziv aplikacije, tekst za verziju laravela i tekst s nazivom autora aplikacije. Također, s linijom *@if (Route::has('login'))* provjera se ukoliko postoji ruta *'login'* te ukoliko ona postoji slijedi još provjera ako je korisnik prijavljen u aplikaciju ili ne koja se izvršava s naredbom *@auth*. Ukoliko je korisnik prijavljen u aplikaciju na *welcome* stranici prikaz će se gumb „*Dashboard*“ koji vraća na rutu *'dashboard'*, a ako korisnik nije prijavljen na *welcome* stranici će se prikazati gumb za „*Log in*“ koji vodi na rutu *'login'*. Detaljni kod prikazan je na Slici 7. i Slici 8, dok slika uređene *welcome* stranice prikazana je na Slici 9.

```
<body class="antialiased">
    <div class="relative sm:flex sm:justify-center sm:items-center min-h-screen bg-dots-darker bg-center bg-gray-100 dark:bg-dots-lighter
    dark:bg-gray-900 selection:bg-red-500 selection:text-white">
        @if (Route::has('login'))
            <div class="sm:fixed sm:top-0 sm:right-0 p-6 text-right z-10">
                @auth
                    <a href="{{ url('/dashboard') }}" class="font-semibold text-gray-600 hover:text-gray-900 dark:text-gray-400 dark:hover:text-white
                    focus:outline focus:outline-2 focus:rounded-sm focus:outline-red-500">Dashboard</a>
                @else
                    <a href="{{ route('login') }}" class="font-semibold text-gray-600 hover:text-gray-900 dark:text-gray-400 dark:hover:text-white
                    focus:outline focus:outline-2 focus:rounded-sm focus:outline-red-500">Log in</a>
                @endauth
            </div>
        @endif
```

Slika 7. Prikaz koda za stranicu welcome 1.dio

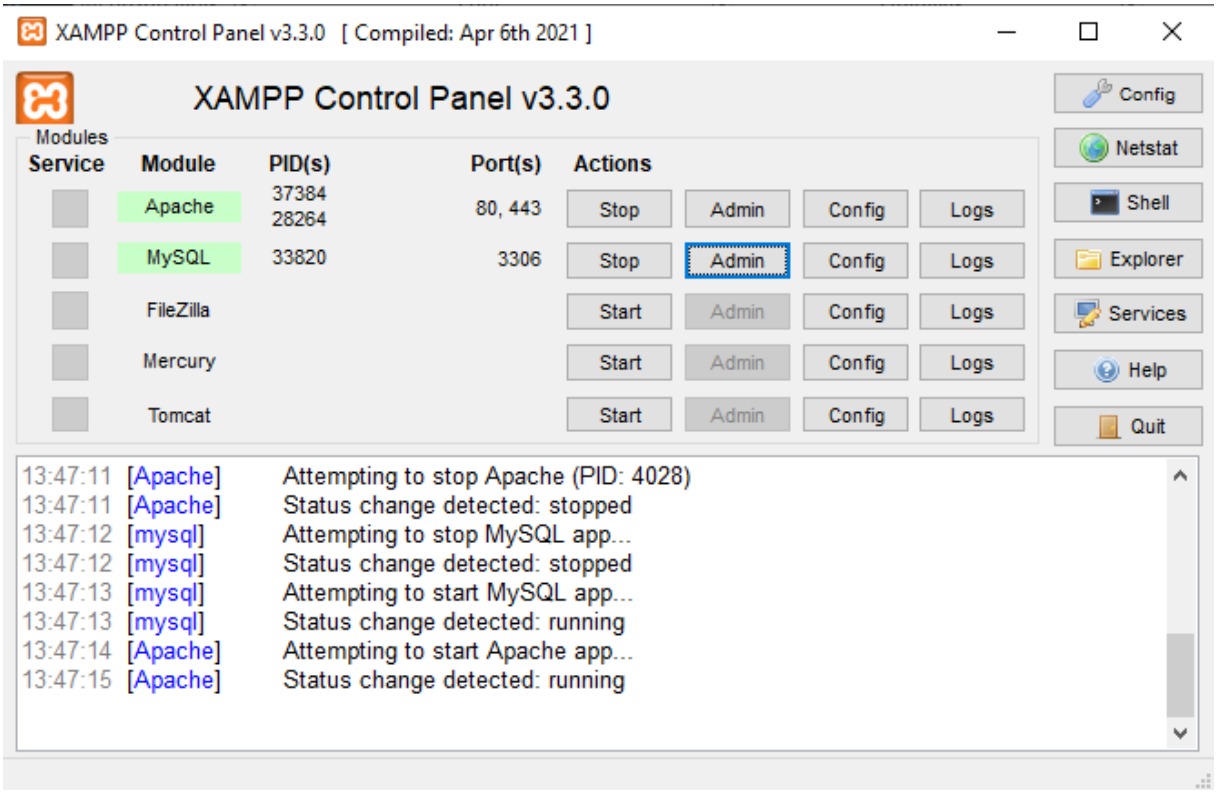
```
<div class="max-w-7xl mx-auto p-6 lg:p-8">
    <div class="mt-16">
        <div class="flex justify-center ">
            <div class="scale-100 p-6 bg-white dark:bg-gray-800/50 dark:bg-gradient-to-bl from-gray-700/50 via-transparent dark:ring-1
            dark:ring-inset dark:ring-white/5 rounded-lg shadow-2xl shadow-gray-500/20 dark:shadow-none flex motion-safe:hover:scale-[1.01]
            transition-all duration-250 focus:outline focus:outline-2 focus:outline-red-500">

                <h1>APLIKACIJA ZA UPISE U VIŠU GODINU STUDIJA<h1></br>

            </div>
        </div>
    </div>
    <div class="flex justify-center mt-16 px-0 sm:items-center sm:justify-between">
        <div class="ml-4 text-center text-sm text-gray-500 dark:text-gray-400 sm:text-right sm:ml-0">
            <p>©AnaFabac</p>
        </div>
        <div class="ml-4 text-center text-sm text-gray-500 dark:text-gray-400 sm:text-right sm:ml-0">
            Laravel v{{ Illuminate\Foundation\Application::VERSION }} (PHP v{{ PHP_VERSION }})
        </div>
    </div>
</div>
</div>
</body>
```

Slika 8. Prikaz koda za stranicu welcome 2.dio

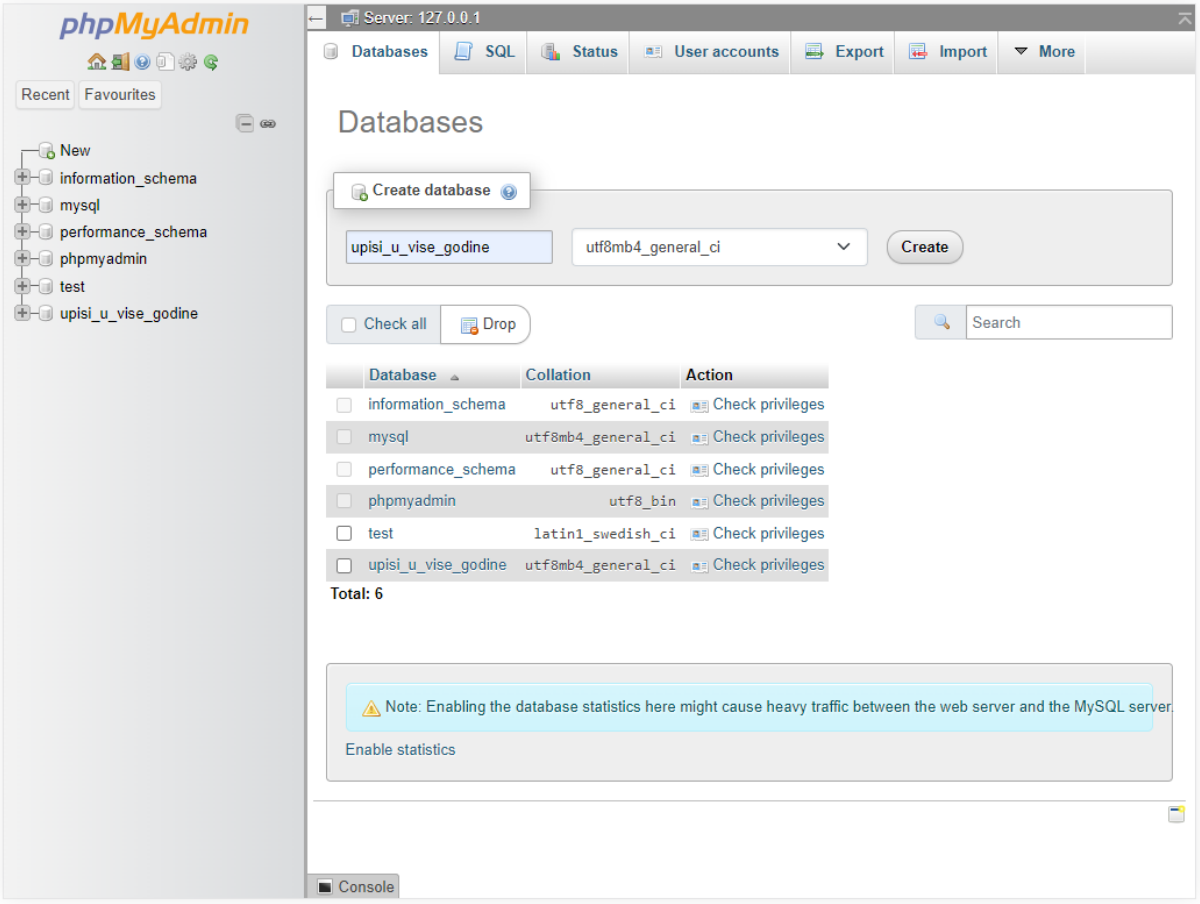


Slika 9. Prikaz stranice welcome u web pregledniku

Prije samog početka kreiranja tablica potrebno je laravel aplikaciju povezati s bazom podataka te za upravljanjem bazom koristit će se *XAMPP* aplikacija. Potrebno je otvoriti *XAMPP* aplikaciju te pokrenuti module *Apache* i *MySQL* i kliknuti na gumb *„Admin“* kod *MySQL* kako bi se otvorila baza u web pregledniku. Nakon otvaranja potrebno je kliknuti na gumb *„New“* i dodani novu bazu s naslov *upisi_u_vise_godine* i kliknuti na gumb „Create“. Nakon kreiranje baze potrebno ju je povezati s Laravel projektnom tako da se u *.env* datoteci promijeni *DB_DATABASE= laravel* u *DB_DATABASE= upisi_u_vise_godine.*



Slika 10. Prikaz sučelja XAMPP aplikacije

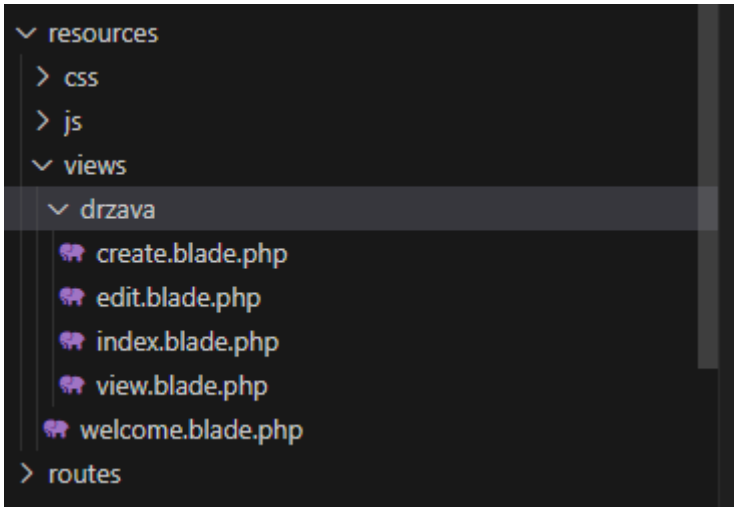


Slika 11. Prikaz pokrenute aplikacije za baze u web browseru

```
10
11  DB_CONNECTION=mysql
12  DB_HOST=127.0.0.1
13  DB_PORT=3306
14  DB_DATABASE=upisi_u_vise_godine
15  DB_USERNAME=root
16  DB_PASSWORD=
17
```

Slika 12. Prikaz .env datoteke

Za svaku tablicu koja je opisana u modelu podataka potrebno je kreirati u Laravelu datoteku za kreiranje novih podataka, uređivanje podataka, prikaz svih podatak iz tablice te prikaz samo odabranog retka iz tablice. Što znači, da za tablicu država potrebno je kreirati mapu *drzava* unutar mape *views* te u mapi *drzava* kreirati datoteke *create.blade.php, edit.blade.php index.blade.php* i *view.blade.php*.



Slika 13. Prikaza mape drzava

Nakon kreiranje datoteka potrebno je i kreirati samu tablicu *drzava* na način da se u terminalu upiše naredba *php artisan make:migration create_drzava_table* čime se kreira nova migracijska datoteka unutar mape *database* pa *migrations*. Otvaranjem datoteke vidljive su funkcije *up()* i *down()* koje se koriste za stvaranje ili brisanje tablice iz baze. Također, u funkciji *up()* je već definira funkcija *Schema::create('drzava', function (Blueprint $table)* koja sadrži naziv tablice koji je upisan prilikom pokretanja naredbe te funkcija sadrži *$table->id()* koji je ID tablice i *$table->timestamps()* koji služi za pohranu vremena kada je kreiran novi podatak u tablici. Za potrebe tablice *drzava* u liniji za kreiranje ID potrebno je dodati naziv *id_drzave* te ostaviti tip podataka *id* koji je zapravo tip podataka *bigIncrements* i stvara stupac kao primarni ključ tablice. Osim toga, potrebno je dodati novi stupac *naziv_drzave* koji je tipa *string*. Nakon dodavanja u funkciju *up()* potrebno je spremiti promjene te u terminalu pokrenuti funkciju *php artisan migrate* kojom se izvršavaju sve nove migracije u bazi čime se i u

konačnici kreira tablica *drzava* u kojoj sada se mogu spremiti podaci. Prikaz cijele migracijske datoteke za tablicu *drzava* prikazana je na Slici 14.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('drzava', function (Blueprint $table) {
            $table->id('id_drzave');
            $table->string('naziv_drzave');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('drzava');
    }
};

```

Slika 14. Prikaz migracijske datoteke za tablicu drzava

Za tablicu *drzava* potrebno je kreirati upravitelja *DrzavaController.php* koji se kreira u terminalu pomoću funkcije *php artisan make:controller DrzavaController –resource* i nalazi se unutar direktorija *app/Http/Controllers*. Upravitelji (eng. „controller“) u laravelu služe za grupiranje logike upravljanja i rukovanjem zahtjevima u jednoj klasu kako bi se olakšalo programerima te oni sadrži metode za kreiranje, čitanje, ažuriranje, pohranu i brisanje podataka iz tablice [7]. Osim upravitelja, za tablicu *drzava* potrebno je kreirati i model *Drzava.php* pomoću funkcije *php artisan make:Model Drzava* koji se nalazi u direktoriju *app/Models*. U laravelu modeli komuniciraju s bazom podataka na način da model dohvaća podatke iz baze podataka, nakon čega model prosljeđuje podatke upraviteljima. Model služi i kako bi se definirale veze između tablica u bazi podataka [8].

U modelu *Drzava.php*, potrebno je definirati naziv tablice s kojom se model povezuje pomoću linije *protected $table='drzava'*, primarni ključ tablice pomoću linije *protected $primaryKey= 'id_drzave'* te potrebno je definirati ostale stupce tablice koje se mogu uređivati s *protected*

$fillable=['id_drzave','naziv_drzave']. Kako je tablica *drzava* povezana s tablicom *zupanija* na način da država ima više županija dok jedna županija pripada samo jednoj državi potrebno je definirati i tu vezu u modelu. Za stvoriti vezu potrebno je kreirati novu funkciju *zupanija()* u kojoj definiramo *One to Many* vezu s funkcijom *hasMany(Zupanija:class)* čime se definiralo da država ima više županija. Cijeli kod u modelu *Drzava.php* prikazan je na Slici 15.

upisiuvisegodine-app > app > Models > Drzava.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

11 references | 0 implementations
class Drzava extends Model
{
    0 references
    protected $table='drzava';
    0 references
    protected $primaryKey= 'id_drzave';
    0 references
    protected $fillable=['id_drzave','naziv_drzave'];
    use HasFactory;

    0 references | 0 overrides
    public function zupanija()
    {
        return $this->hasMany(Zupanija::class);
    }
}

```

Slika 15. Prikaz modela Drzava.php

Nakon kreiranja modela potrebno ga je povezati s upraviteljem tako da se u upraviteljskoj datoteci doda putanja modela datoteke koja je use *App\Models\Drzava.* Prva funkcija koja se definira u upravitelju je *index()* u kojoj se pomoću modela *Drzava* i metode *all()* dohvaćaju podaci o svim državama koji su u bazi podataka te spremaju se u varijablu *$drzava*. Nakon čega, s metodom *return()* vraća *'drzava.index'* uz pomoću metode *view()* koja zapravo poziva prethodno stvorenu datoteku *index.blade.php* u mapi *drzava.* Podaci spremljeni u varijabli *$drzava* pomoću metode *with()* šalju se u *'drzava.index'* kako bi se mogli prikazati. Druga funkcija koja se definira je *create()* koja vraća metodu *view('drzava.create*') kako bi se podaci prikazali po *drzava.create.blade.php* datoteci koja se koristi prilikom unosa novih podataka u bazu. Dok funkcija *store()* definira način spremanja unošenih podataka u bazu podataka na način da se dohvaća HTTP zahtjev s *Request $request.* Svi podaci iz zahtjeva se dohvaćaju s *$request->all()* i spremaju u varijablu *$input* nakon čega se poziva model *Drzava* i metoda

create() kojoj se prosljeđuje *$input* kako bi se kreirao novi stupac u tablici *drzava* i spremili podaci. Nakon uspješnog spremanja podataka korisnika funkcija *redirect()* preusmjeri natrag na rutu *'drzava'* koja zapravo vraća *view('drzava.index').* Prikaz koda za funkcije *indeks(), create()* i *store()* je vidljiv na Slici 16.

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;
use App\Models\Drzava;


2 references | 0 implementations
class DrzavaController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    0 references | 0 overrides
    public function index()
    {
        $drzava=Drzava::all();
        return view('drzava.index')->with('drzava',$drzava);
    }

    /**
     * Show the form for creating a new resource.
     */
    0 references | 0 overrides
    public function create()
    {
        return view('drzava.create');
    }

    /**
     * Store a newly created resource in storage.
     */
    0 references | 0 overrides
    public function store(Request $request):RedirectResponse
    {
        $input = $request->all();
        Drzava::create($input);
        return redirect('drzava')->with('flash_message', 'Dodana nova država!');
    }
```

Slika 16. Prikaz kontreolera DrzavaController 1.dio

Sljedeća funkcija je *show()* koja se koristi za prikazivanje samo odabranog retka iz tablice te u funkciju se prosljeđuje ID retka koji se želi prikazati. S metodom *find()* pretražuje se baza podataka uz korištenje modela *Drzava* kako bi se pronašao redak kojem ID pripada te se podaci spremaju u varijablu *$drzava*. Nakon čega, se podaci spremljeni u varijabli *$drzava* šalju u

view('drzava.view') pomoću metodom *with()* kako bi se oni prikazali. Na isti način kao i funkcija *show()* definira se i funkcija *edit()* kod koje se podaci šalju na *view('drzava.edit')* kako bi se prikazala forma za uređivanje podataka. Kada se ispuni forma za uređivanje potrebno je promjene spremiti pomoću funkcije *update()* u kojoj se prosljeđuje *Request $request i $id_države* kako bi se dohvatile dobivene promjene i ID retka za koje su promjene napravljene. Kao i kod funkcije *store()* u modelu se traži ID retka kod koje je promjena napravljena te se s metodom *update()* ažuriraju podaci. Nakon ažuriranja podataka korisnika se pomoću funkcije *redirect()* preusmjeri natrag na *view('drzava.index').* Posljednja funkcija je *destroy()* koja se definira se isti način kao i funkcija *show()* ona služi za brisanje podataka odabranog retka iz tablice te za brisanje iz modela se koristi metoda *destroy()* u kojoj se prosljeđuje ID retka kojeg se želi izbrisati. Kao i kod funkcija *store()* i *update()* nakon brisanja podataka se korisnika preusmjeri na *view('drzava.index').* Slika 17. prikazuje kod *za show(), store(), edit(), update() i destroy().*

```php
public function show(string $id_drzave)
{
    $drzava = Drzava::find($id_drzave);
    return view('drzava.view')->with('drzava', $drzava);
}

/**
 * Show the form for editing the specified resource.
 */
0 references | 0 overrides
public function edit(string $id_drzave)
{
    $drzava = Drzava::find($id_drzave);
    return view('drzava.edit')->with('drzava', $drzava);
}

/**
 * Update the specified resource in storage.
 */
0 references | 0 overrides
public function update(Request $request, string $id_drzave):RedirectResponse
{
    $drzava = Drzava::find($id_drzave);
    $input = $request->all();
    $drzava->update($input);
    return redirect('drzava')->with('flash_message', 'Podaci o državi su uspješno promijenjeni!');
}

/**
 * Remove the specified resource from storage.
 */
0 references | 0 overrides
public function destroy(string $id_drzave):RedirectResponse
{
    Drzava::destroy($id_drzave);
    return redirect('drzava')->with('flash_message', 'Država uspješno izbrisana!');
}
```

Slika 17. Prikaz kontreolera DrzavaController 2. .dio

Način prikaza podataka definira se u datotekama koje su pozvane u upravitelju s obzirom na korake koje korisnik radi. Prikaz svih podataka tablice *drzava* definira se u datoteci

index.blade.php koja se nalazi unutar direktorija *resources/views/drzav'*. Kod započinje s naredbom *@extends('main')* koja je dio laravel *blade* predloška i služi za proširivanje već postojećeg rasporeda na novu stranicu. Dok s naredbenom *@section('content')* se ubacuje sadržaj na novu stranicu unutar predviđene sekcije koja je prethodno definirana s naredbom *@yield('content')* u *main.blade.php* datoteci. Nakon toga, slijedi HTML kod za prikaz podatak iz tablice država na način da je korištena oznaka *card* iz boostrap 5 kako bi se sadržaja prikazao unutar okvira.

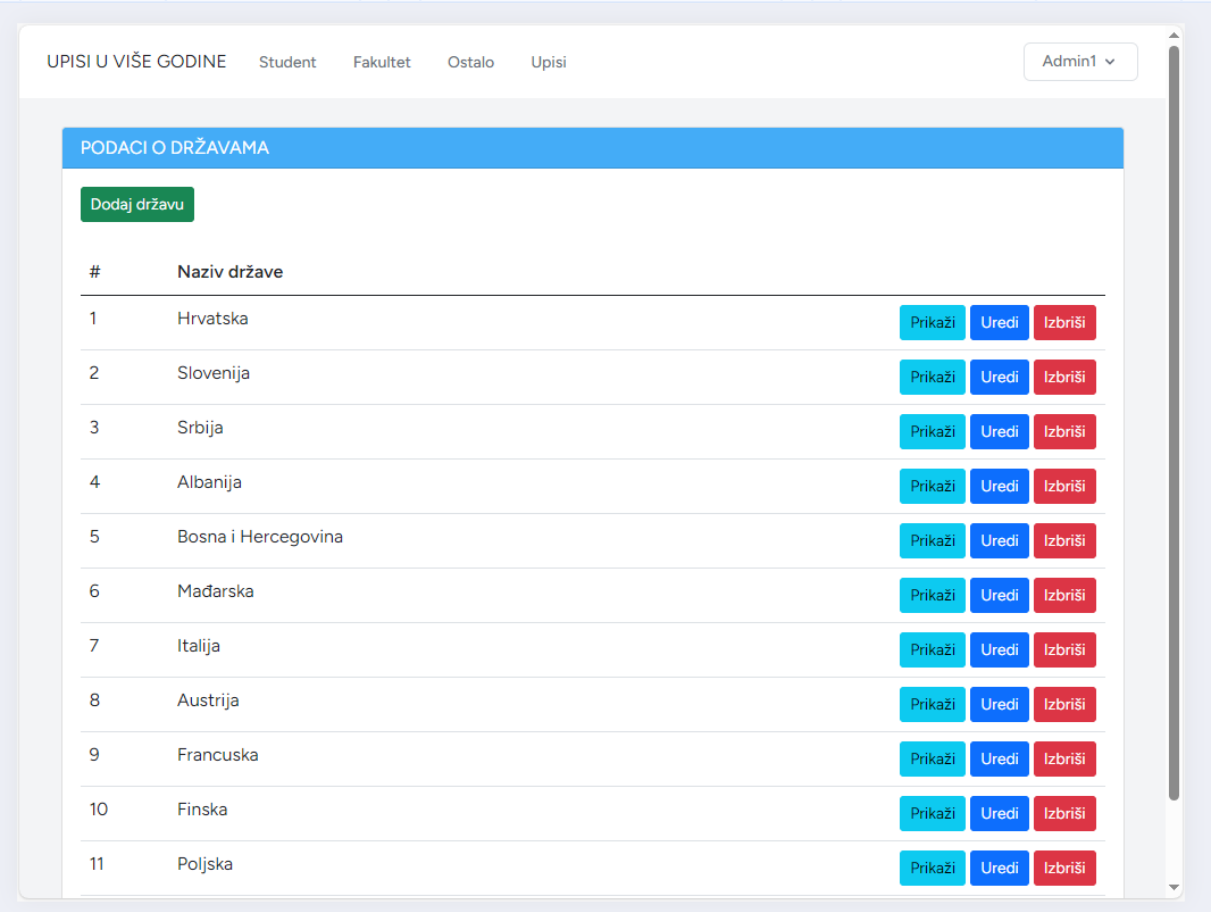
Prije prikaza tablice dodan je gumb „Dodaj državu“ koji vodi na rutu *'drzava/create'* koja je u *DrzavaController.php* s funkcijom *create()* definirano da otvara *create.blade.php* datoteku za prikaz forme za kreiranje novog retka u tablici *drzava*. Podaci se prikazuju u obliku tablice na način da se unutar sekcije *<table>* pa *<thead>* upiše naslovi stupaca dok u sekciji *<tbody>* upisuju redci iz tablice država. Upis podataka u retke vrši se pomoću naredbe *@foreach($drzava as $item)* čime se prolazi kroz svaki redak tablice i za njega ispisuje svaki stupac tako da se prvo napiše ime tablice zatim stupac koji se poziva. Kako se u naredbi *@foreach* definiralo da za tablicu $*drzava* koristi naziv $*item* onda naredba za poziv retka *id_drzave* treba glasiti *{{$item-> id_drzave}}*. Uz svaki redak prikazuje se i gumb „Prikaži“, „Uredi“, „Izbriši“ te klikom na njih korisnika se preusmjeri na drugu rutu ovisno o odabiru. Za svaki od tih gumba ruta se tvori od statičkog djela *'/drzava'* i dinamičkog dijela *$item->id_drzave* kako bi se prikazali, uredili ili izbrisali podaci za odabrani redak iz tablice. Te na kraju potrebno je završiti naredbu *@foreach* s naredbom *@endforeach* te s naredbom *@endsection* označava se kraj naredbe *@section('content).* Kod za *index.blade.php* datoteku je prikazan na Slici 18. i Slici 19., dok na Slici 20. se prikazuje prikaz tablice u web pregledniku.

```
@extends('main')
@section('content')

                <div class="card">
                    <div class="card-header text-white" style="background-color: #44ACF7;">
                        <h2>PODACI O DRŽAVAMA<h2>
                    </div>
                    <div class="card-body">
                        <a href="{{ url('/drzava/create') }}" class="btn btn-success btn-sm" title="Dodaj novu državu">
                            <i class="fa fa-plus" aria-hidden="true"></i> Dodaj državu
                        </a>
                        <br/>
                        <br/>
                        <div class="table-responsive" >
                            <table class="table" >
                                <thead>
                                    <tr>
                                        <th>#</th>
                                        <th>Naziv države</th>
                                        <th></th>
                                </thead>
```

Slika 18. Prikaz koda unutar index.blade.php 1.dio

```
            <tbody>
            @foreach($drzava as $item)
                <tr>
                    <td>{{ $loop->iteration}}</td>
                    <td>{{ $item->naziv_drzave }}</td>
                    <td align="right">
                        <a href="{{ url('/drzava/' . $item->id_drzave) }}" title="Prikaži podatke o državi"><button class="btn btn-info btn-sm">
                            <i class="fa fa-eye" aria-hidden="true"></i> Prikaži</button></a>
                        <a href="{{ url('/drzava/' . $item->id_drzave . '/edit') }}" title="Uredi podatke o državi"><button class="btn btn-primary btn-sm">
                            <i class="fa fa-pencil-square-o" aria-hidden="true"></i> Uredi</button></a>

                        <form method="POST" action="{{ url('/drzava' . '/' . $item->id_drzave) }}" accept-charset="UTF-8" style="display:inline">
                            {{ method_field('DELETE') }}
                            {{ csrf_field() }}
                            <button class="btn btn-danger btn-sm" title="Izbriši državu" onclick="return confirm(&quot;Confirm delete?&quot;)">
                            <i class="fa fa-trash-o" aria-hidden="true"></i> Izbriši</button>
                        </form>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>

</div>
</div>
```

Slika 19. Prikaz koda unutar index.blade.php 2.dio



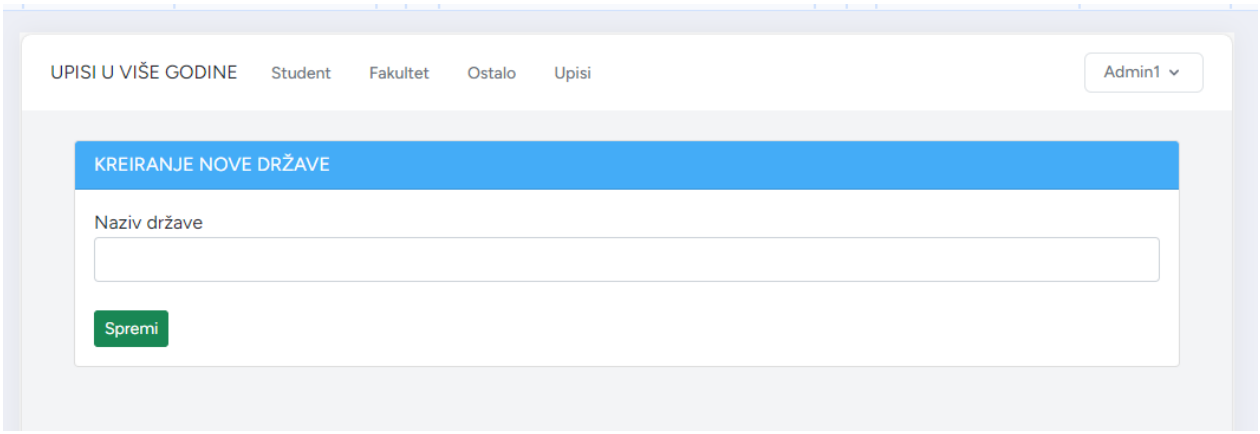
Slika 20. Prikaz tablice država u web pregledniku

Klikom na gumb „Dodaj državu“ korisnika se preusmjerava na rutu *'drzava/create'* koja otvara formu za upis nove države čiji je prikaz definiran u *create.blade.php* datoteci. Kod započinje kao i kod *index.blade.php* datoteke s naredbama *@extends('main')* i *@section('content')* nakon kojih ide HTML kod pa naredba *@stop* kako bi se završila sekcija sa sadržajem. Za kreiranje

novih podataka koriste se sekcija <*form*> u kojoj je potrebno definirati da se koristi HTTP metoda *post* koja služi za slanje podataka koji se šalju funkciji *store()* unutar upravitelja te ona ih sprema u tablicu. Također, kada se kreiraju, mijenjaju ili brišu podaci koristi se *blade* direktiva *{!! csrf_field() !!}* koja generira skriveno polje koje sadrži CSRF token koji se koristi za provjeru da li je zahtjev poslan od strane autentificiranog korisnika time se zaštićuje aplikacija od CSRF napada. Forma za kreiranje se sastoji od naslova i polja za unos podatka te u kodu kod definiranja polja bitno je da atributima *name* i *id* je dodan točan stupac iz tablice kako bi se novi podatak spremio u točan stupac. Što znači ukoliko u polje za unos podataka je namijenjeno za unos naziva države kod definiranja potrebno je postaviti da *name="naziv_drzave"* i *id="naziv_drzave"* kako bi se mogao identificirati koji je ulazni element i u koji stupac tablice se sprema. Kod je prikazan na Slici 21., dok prikaz na webu je prikazan na Slici 22.

```
@extends('main')
@section('content')

<div class="card">
  <div class="card-header text-white" style="background-color: #44ACF7;">KREIRANJE NOVE DRŽAVE</div>
  <div class="card-body">

    <form action="{{ url('drzava') }}" method="post">
      {!! csrf_field() !!}
      <label>Naziv države</label></br>
      <input type="string" name="naziv_drzave" id="naziv_drzave" class="form-control"></br>
      <button class="btn btn-success btn-sm"  value="Save" title="Spremi podatke" ><i class="fa fa-eye" aria-hidden="true"></i> Spremi</button>

    </form>

  </div>
</div>

@stop
```

Slika 21. Prikaz koda unuter create.blade.php datoteke



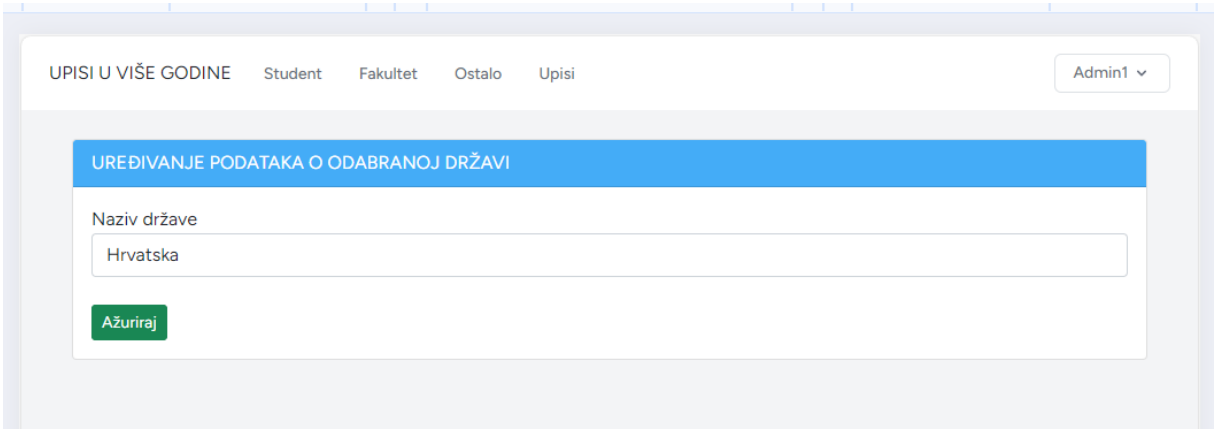
Slika 22. Prikaz forme za unos nove države

Klikom na gumb *„Uredi“* u prvom retku tablice se korisnika preusmjerava na rutu */'drzava/{id_drzave}/edit'* koja je u upravitelju *DrzavaController.php* u funkciji *edit()* definirana da otvara *edit.blade.php* datoteku za uređivanje podataka za odabrani redak iz tablice. Datoteka je uređena po istom principu kao i *create.blade.php* na način da se koriste naredbe *@extends('main'), @section('content') i @stop.* Dok u HTML kodu kod input polja dodaje se još jedan atribut *value* koji se koristi za popunjavanje polja za unos sa vrijednosti koja je prethodna bila spremljena što znači ako je *value=“ $drzava->id_drzave“* onda se u polju

prikazuje vrijednost *$drzava->id_drzave*. Kod za *edit.blade.php* je prikazan na Slici 23., dok prikaz na webu je prikazan na Slici 24.

```
@extends('main')
@section('content')

<div class="card">
  <div class="card-header text-white" style="background-color: #44ACF7;">UREĐIVANJE PODATAKA O ODABRANOJ DRŽAVI</div>
  <div class="card-body">

      <form action="{{ url('drzava/' .$drzava->id_drzave) }}" method="post">
        {!! csrf_field() !!}
        @method("PATCH")
        <label>Naziv države</label></br>
        <input type="string" name="naziv_drzave" id="naziv_drzave" value="{{$drzava->naziv_drzave}}" class="form-control"></br>
        <button class="btn btn-success btn-sm"  value="Update" title="Ažurirajte podatke" ><i class="fa fa-eye" aria-hidden="true"></i> Ažuriraj</button>


    </form>

  </div>
</div>

@stop
```

Slika 23. Prikaz koda unutar edit.blade.php datoteke

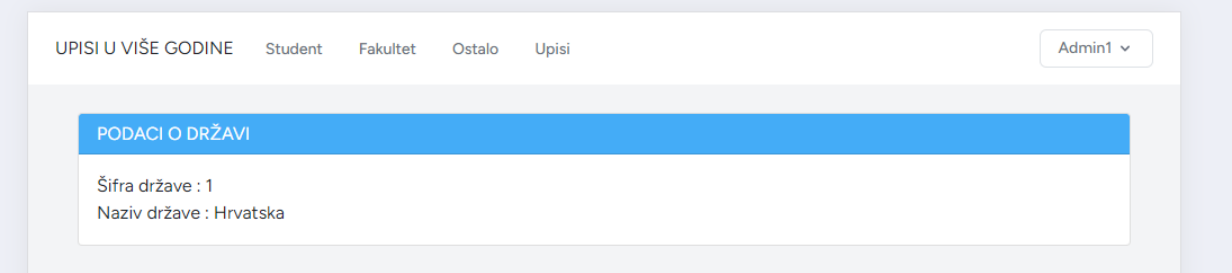


Slika 24. Prikaz forme za uređeivanja podataka o državi

Klikom na gumb *„Prikaži“* u prvom retku tablice korisnika se preusmjerava na rutu /'d*rzava/{id_drzave}'* koja je u upravitelju *DrzavaController* u funkciji *show()* definirana da otvara *view.blade.php* datoteku za prikaz samo odabranog retka iz tablice. Datoteka je uređena po istom principu kao i *index.blade.php* na način da se koriste naredbe *@extends('main')*, *@section('content')* nakon kojih ide HTML kod pa naredba *@stop* kako bi se završila sekcija sa sadržajem. U HTML kodu za svaki stupac tablice potrebno je posebno pozvati po principu naziv tablice pa onda naziv atributa čime bi naredba za atribut *id_drzave* glasila *$drzava->id_države*. Kod je prikazan na Slici25., dok prikaz na webu je prikazan na Slici 26.

```
@extends('main')
@section('content')


  <div class="card">
    <div class="card-header text-white" style="background-color: #44ACF7;"><h2>PODACI O DRŽAVI</h2></div>
    <div class="card-body">



        <!-- <h5 class="card-title">PODACI O DRZŽAVI</h5> -->
        <p class="card-text">Šifra države : {{ $drzava->id_drzave }}</p>
        <p class="card-text">Naziv države : {{ $drzava->naziv_drzave }}</p>


    </div>
  </div>

  @stop
```

Slika 25. Prikaz koda u datoteci view.blade.php



Slika 26. Prikaz odabranog retka u talici drzava

Cijeli proces od kreiranje tablice, dodavanje stupaca, migracija, kreiranje modela i upravitelja te uređivanja datoteka za prikaz podataka je jednak za tablice *drzavljanstvo*, *sveuciliste*, *razina*, *strucna_sprema*, *obrazovanje*, *zanimanje* i *polozaj_u_zanimanju* kao za tablicu *drzava* čiji je proces prethodno opisan.

3.1. TABLICE SLABOG TIPA ENTITETA

Kod tablica koje su od slabog tipa entiteta proces kreiranja tablice, migracija i kreiranje modela, upravitelja i datoteka za prikaz je jednak kao i kod tablica koje su od jakog tipa entiteta, no u kodu postoje manje razlike. Tip entiteta *ZUPANIJA* je slabi tip entiteta od tipa entiteta *DRZAVA* što znači da tablica *zupanija* sadrži vanjski ključ *id_drzave*. Kod kreiranje tablice u migracijsku datoteku potrebno je dodati vanjski ključ na način da se prvo definira stupac tablice $table-> *integer('id_drzave_zupanija')* nakon čega je potrebno definirati da se radi o vanjskom ključu pomoću metode *foreign()*. Osim toga, potrebno je definirati da se vanjski ključ referencira na *id_drzave* iz tablice *drzava* i potrebno je postaviti *onDelete* na *cascade* pošto se radi o vezi između slabe i jake tablice. Čime se dobiva da cijela naredba treba glasiti *$table->foreign('id_drzave_zupanija')->references('id_drzave')->on('drzava')->onDelete('cascade').* Prikaz cijele migracijske datoteke za *zupaniju* je prikazan na Slici 27.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('zupanija', function (Blueprint $table) {
            $table->id('id_zupanije');
            $table->string('naziv_zupanije');
            $table->integer('id_drzave_zupanija');
            $table->foreign('id_drzave_zupanija')->references('id_drzave')->on('drzava')->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('zupanija');
    }
};
```

Slika 27. Prikaz migarijkse datote za tablicu zupanija

U modelu *Zupanija.php* potrebno je dodati vezu s tablicom *drzava* koja se radi na način da se kreira nova *public* funkcija *drzava()* unutar modela. U funkciji potrebno je definirati vezu između tablica pomoću metode *belongsTo()*, koja označava da županija pripada državi te u metodi se upisuje model *Drzava::class* i vanjski ključ na koji se referencira na tablicu *drzava* u ovom slučaju je to *id_drzave_zupanija* unutar tablice *zupanija*. Prikaz modela *Zupanija.php* je na Slici 28.

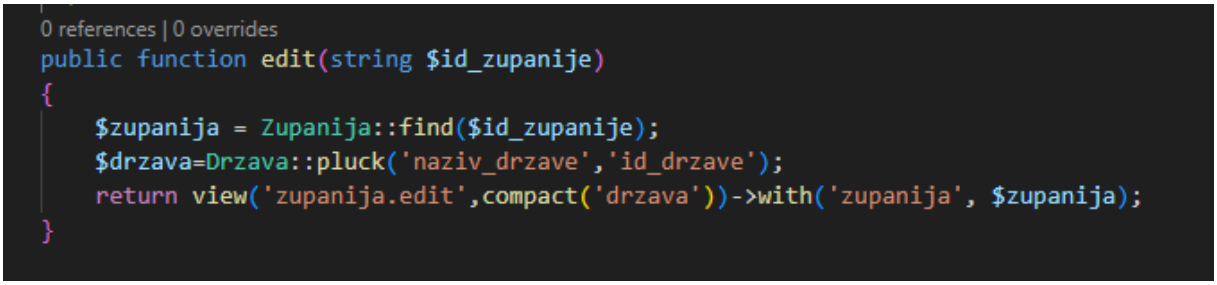
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasOne;


12 references | 0 implementations
class Zupanija extends Model
{
    0 references
    protected $table='zupanija';
     0 references
    protected $primaryKey= 'id_zupanije';
    0 references
    protected $fillable=['id_zupanije','naziv_zupanije','id_drzave_zupanija'];
    use HasFactory;

    0 references | 0 overrides
    public function drzava()
    {

        return $this->belongsTo(Drzava::class,'id_drzave_zupanija');
    }
    0 references | 0 overrides
    public function opcina()
    {

        return $this->hasMany(Opcina::class);
    }
}
```

Slika 28. Prikaz modela Zupanija.php

Dok u upravitelju *ZupanijaController.php* u funkcije *create()* i *edit()* potrebno je dodati liniju *$drzava=Drzava::pluck('naziv_drzave','id_drzave');* s kojom se u varijablu *$drzava* spremaju dohvaćene vrijednosti metodom *pluck()* iz modela *Drzava*. Također, prilikom vraćanja metode *view()* potrebno je u njoj i uključiti varijablu *$drzava* pomoću metode *compact()* kako bi se u datoteci za prikaz mogli pozvati i podaci iz tablice *drzava*. Prikaz funkcije *create()* prikazana je na Slici 29., dok funkcije *edit()* na Slici 30.

```php
0 references | 0 overrides
public function create()
{
    $drzava=Drzava::pluck('naziv_drzave','id_drzave');
    return view('zupanija.create',compact('drzava'));
}
```

Slika 29. Prikaz funkcije create() u ZupanijuController.php



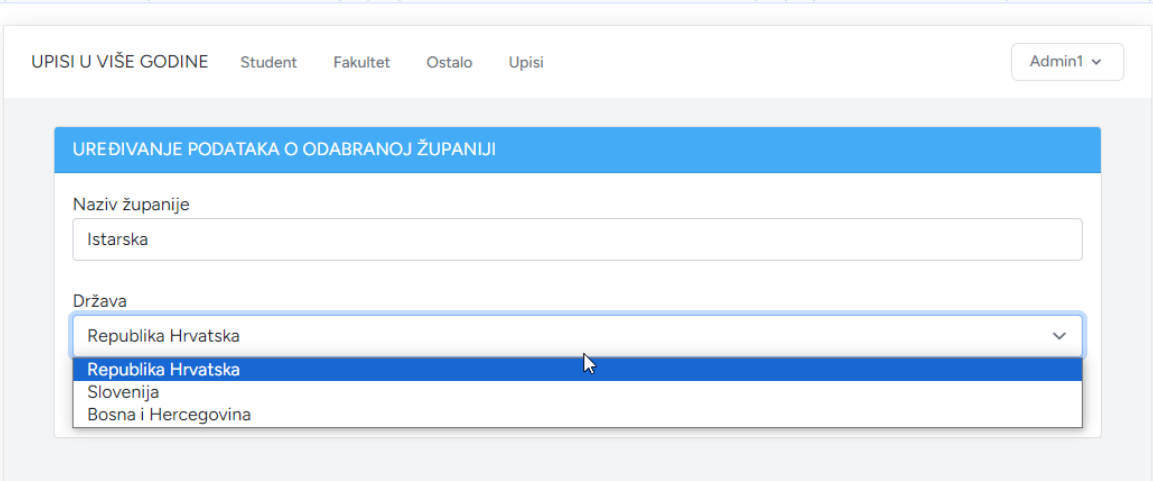
Slika 30. Prikaz funkcije edit() u ZupanijuController.php

Kod datoteke *create.blade.php* koja se nalazi u mapi *zupanija* kod za polje za unos podataka o državi razlikuje se od normalnog unosa pošto je potrebno da se prikaže padajući izbornik s nazivima država. Za kreiranja padajućeg izbornika potrebno je unutar sekcije *<select>* definirati petlju koja prolazi kroz sve države na način da se unutar *@foreach* prosljeđuje varijabla *$država* koja je prethodno definirala u funkciji *create()* te s metodom *compact()* proslijeđena u datoteku. Kako su u upravitelju u varijabli *$drzava* spremljeni *id_drzave* i *naziv_drzave* u uvjetu for petlje potrebno ih je odvojiti te naredbeni redak za to je *$drzava as $id=>$name*. Unutar for petlje u sekciji *<option>* definira se da je *value* jednak varijabli *$id* pošto se on treba spremati u stupac *id_drzave_zupanija*, dok se ima države koje je sadržano u varijabli *$name* ispisuje na ekranu. Cijeli kod za *create.blade.php* je vidljiv na Slici 31. Po istom principu vanjski ključ se definira i u datoteci *edit.blade.php* uz dodatan uvjet da se u polju za unos prikazuje prethodno odabrani izbor. Uvjet se radi uz naredbu *@if* koja provjerava ukoliko je ime države odgovara trenutnom izboru ili prethodnom izboru te ukoliko odgovara onda se on prikazuje u polje za unos. Za provjeru prethodnog odabira koristi se funkcija *old()*. Primjena funkcije *old()* i cijela provjera prikazana je na Slici 32.

Slika 31. Kod unutar datoteke create.blade.php

```
<label>Država</label></br>
  <select  name='id_drzave_zupanija' id="id_drzave_zupanija" value="{{$zupanija->id_drzave_zupanija}}" class="form-control select2">
  @foreach ($drzava as $id => $naziv)
  <option value="{{ $id }}"  @if($zupanija->drzava->naziv_drzave == $naziv || old('naziv') == $naziv) selected  @endif>{{ $naziv }}
  </option>
  @endforeach
  </select></br>
```

Slika 32. Prikaz provjere i korištenje funkcije old() u edit.blade.php

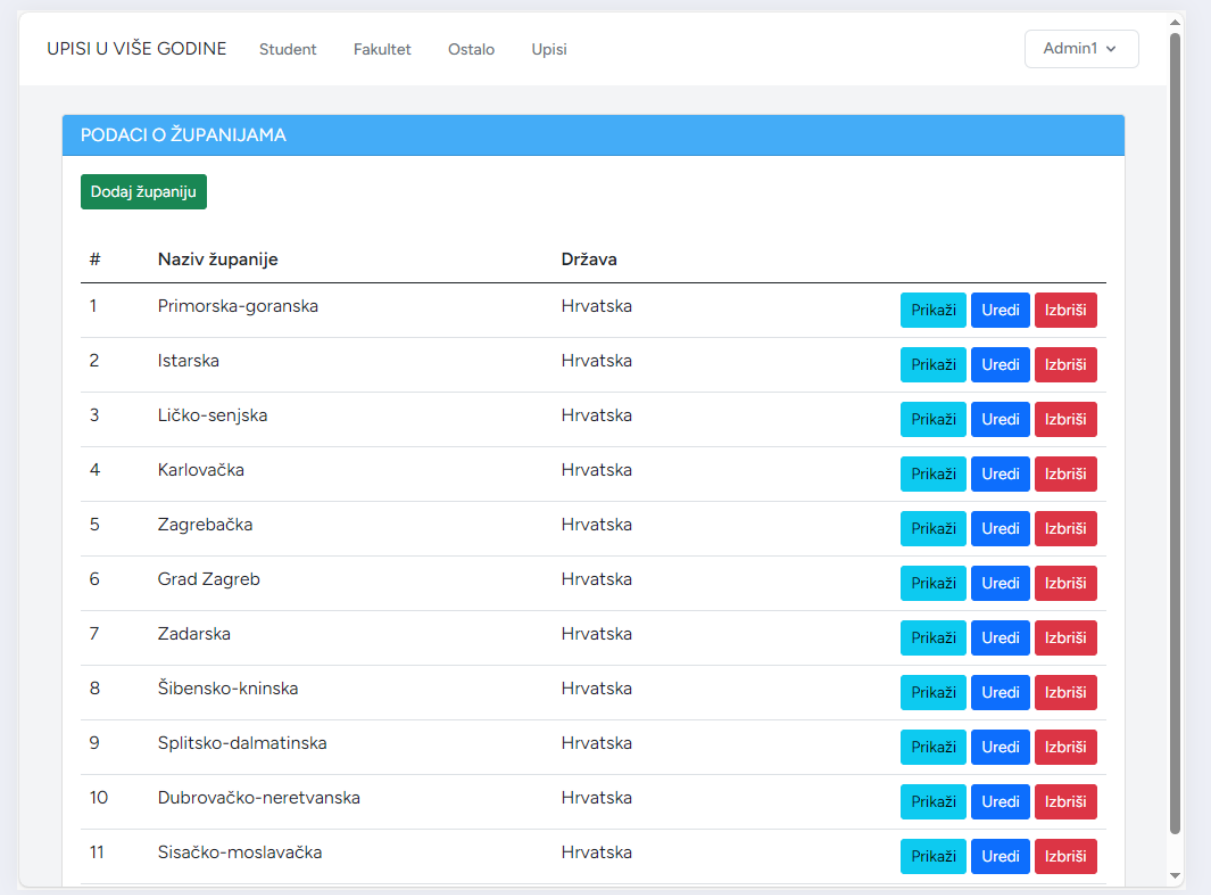


Slika 33. Prikaz padajućeg izbornika kod uređivanja podataka

Dok u datotekama *indeks.blade.php* i *view.blade.php* potrebno je prilagoditi način pozivanja vanjskog ključa da bi se korisniku ispisao naziv države, a ne ID. Način na koji radi je da se pozove tablica *zupanija* te nakon nje pozove tablica *drzava* i iz tablice odabere stupac *naziv_drzave* čime se korisniku prikaže *naziv_drzave* čiji je *id_drzave* iz tablice *drzava* jednak *id_drzave_zupanija* iz tablica *zupanija*. Takav način prikaza je moguć ukoliko su dobro definirane veze između tablice *zupanija* i tablice *drzava* u modelu. Na Slici 34. prikazuje se kod u *view.blade.php* datoteci za tablicu *zupanija*.

```
@extends('main')


@section('content')


<div class="card">
  <div class="card-header text-white" style="background-color: #44ACF7;">PODACI O ODABRANOJ ŽUPANIJI</div>
  <div class="card-body">

        <p class="card-text">Šifra županije : {{ $zupanija->id_zupanije }}</p>
        <p class="card-text">Naziv županije : {{ $zupanija->naziv_zupanije}}</p>
        <p class="card-text">Država : {{ $zupanija->drzava -> naziv_drzave}}</p>

  </div>

    </hr>

  </div>
@stop
```

Slika 34. Prikaz koda u view.blade.*php datoteci za tablicu zupanija*



Slika 35. Prikaz podataka o županiji u web pregledniku

Promjene koje su prethodno pokazane za tablicu *zupanija* potrebno je i napraviti i za tablice *kolegij*, *opcina*, *mjesto*, *modul*, *fakultet*, student, *uzdrzavatelj* i upisi. Što se tiče tablice *kolegij_student* osim vanjskih ključeva ona sadrži i atribute *broj_upisa_na_kolegiju*, *ocjena*, *akademska_godina* i *datum_polaganja* te primarni ključ *id_kolegij_student* koji se koristi zbog jednostavnosti poziva, ažuriranja i brisanja stupaca u tablici kada postoje atributi u agregiranoj tablici pa se i ona se tretira kao normalna tablica.

3.2. PRIKAZ RAZLIČITIH TIPOVA PODATAKA UNUTAR TABLICA

U tablici *kolegij* postoji stupac *obavezan* koji služi za informaciju ukoliko je kolegij obavezan na studiju ili je izborni, te postoji i stupac *semestar* koji služi za izbor semestra u kojem se kolegi održava. Za tablicu *kolegij* u njenoj migracijskoj datoteci potrebno je definirati da stupac *obavezan* je tipa *bool* dok za stupac *semestar* da je tipa *enum* za kojega se očekuju vrijednosti *['I','II','III','IV','V','VI']*. Slika 35. prikazuje migracijsku datoteku za tablicu *kolegij*.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('kolegij', function (Blueprint $table) {
            $table->id('id_kolegija');
            $table->string('naziv_kolegija');
            $table->integer('ECTS_kolegija');
            $table->boolean('obavezan');
            $table->string('preduvjet')->nullable();
            $table->enum('semestar',['I','II','III','IV','V','VI']);
            $table->boolean('ulazi_u_prosjek');
            $table->unsignedBigInteger('id_godine_kolegij');
            $table->foreign('id_godine_kolegij')->references('id_godine')->on('godina')->onDelete('restrict');
            $table->unsignedBigInteger('id_modula_kolegij');
            $table->foreign('id_modula_kolegij')->references('id_modula')->on('modul')->onDelete('restrict');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('kolegij');
    }
};
```

Slika 36. Prikaz migracijske datoteke za tablicu kolegij

U upravitelju *KolegijController.php* u funkcije *create()* i *edit()* potrebno je definirati novu varijablu *$semestar* kojoj se dodaju vrijednosti *['I','II','III','IV','V','VI']* kako bi se u datotekama za prikaz mogao napraviti padajući izbornik. Također, prilikom vraćanja metode *view()* potrebno je i u njoj uključiti varijablu *$semestar* pomoću metode *compact()* kako bi se u datotekama za prikaz mogle pozvati vrijednosti koje su dodane. Prikaz funkcije *create()* u *KolegijController.php* je prikazan na Slici 37.

```php
public function create()
{
    $godina=Godina::pluck('naziv_godine','id_godine');
    $modul=Modul::pluck('naziv_modula','id_modula');
    $semestar=['I','II','III','IV','V','VI'];
    return view('kolegij.create', compact('godina', 'modul','semestar'));
}
```

Slika 37. Prikaz funkcije create() u KolegijController.php

U datoteci *create.blade.php* koja se nalazi u mapi *kolegij* kod za polje za unos podataka o semestru na koji se kolegij održava potrebno je prilagoditi tako da se prikaže padajući izbornik. Izrada je slična kao i kod tablice *zupanija* samo što za stupac *semestar* u *@foreach* prosljeđuje se varijabla *$semestar* koja je prethodno definirala u funkciji *create()* te s metodom *compact()* proslijedila se u datoteku *create.blade.php* te u uvjet for petlje potrebno je postaviti *$semestar as $name.* Cijeli kod za stupac *semestar* prikazan je na Slici 38. Dok u *edit.blade.php* datoteci za prikaz semestra se isto koristi for petlja uz *@if* naredbu kao bi se provjerio koji izbor treba prikazati u polju za unos semestra. Naredba funkcionira na način da provjerava ukoliko je vrijednosti semestra odgovara trenutnom izboru ili prethodnom izboru te ukoliko da onda se on prikazuje u polje za unos semestra. Cijela linija za provjeru, uz funkciju *old()* koja provjera koji je prethodni izbor, glasila bi *@if($kolegij->semestar==$naziv || old('semestar')==$naziv) selected @endif.* Kod za uređivanje stupca *semestar* je prikazan na Slici 39., dok prikaz izgleda na web pregledniku je na Slici 42.

```
<select class='form-control select2' name='semestar' id="semestar">
   @foreach ($semestar as $naziv)
    <option value="{{$naziv}}" selected="" >{{$naziv}}</option>
  @endforeach
</select></br>
```

Slika 38. Prikaz koda za stupac semestar tablice kolegij u create.blade.php

```
<select class='form-control select2' name='semestar' id="semestar" value="{{$kolegij->semestar}}">
  @foreach ($semestar as $naziv)
  <option value="{{ $naziv }}"  @if($kolegij->semestar == $naziv || old('semestar') == $naziv) selected  @endif>{{ $naziv }}
  </option>
  @endforeach
</select></br>
```

Slika 39. Prikaz koda za stupac semestar tablice kolegij u edit.blade.php

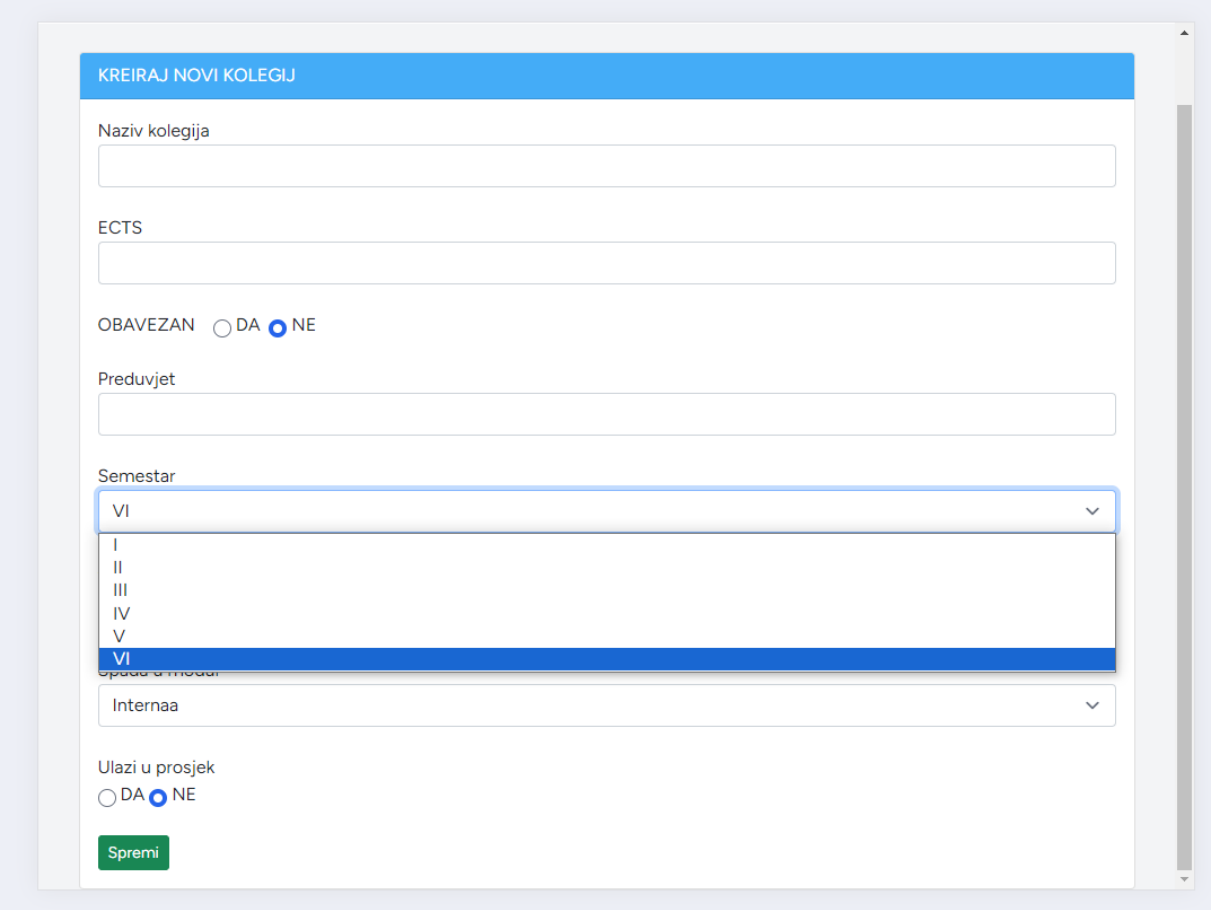
Također, potrebno je prilagoditi izgled polja za stupac *obavezan* na način da se prikazuje *ratio* gumba koje korisnik može odabrati s obzirom na njegov odgovor. U *create.blade.php* u *input* sekciji potrebno je postaviti da *je type=“radio*“ te da je njegov *value=1* i postaviti da je *checked* nakon toga u sekciji *label* upisati *DA* čime se dobiva prikaz *radio* gumba s tekstom *DA* do njega te ukoliko ga korisnik označi on vraća *true* te vrijednost *1* se sprema u stupac *obavezan*. Isti postupak je potrebno napraviti da bi se prikazao gumb s tekstom *NE* uz postavljanje *value=0,* kao što je i prikazano na Slici 40. Naravno potrebno je urediti i način uređivanja *ratio* gumba u *edit.blade.php* datoteci tako da kada korisnik uđe u formu za uređivanje ostaje označeno prethodno odabran odabir. Kao i prethodno koristi se funkcija *old()* s kojom se dohvaća prethodni odabir te nakon toga se provjerava ukoliko odabir je jednak *1*, ako je onda se označuje *DA* ako nije provjera se ako je odabir jednak *0* te ukoliko je označuje se *NE*. Provjere u kodu za *ratio* gumb prikazane su na Slici 41., dok prikaz izgleda na web pregledniku je prikazan na Slici 42.

```
label>OBAVEZAN </label>   
<input class="form-check-input" type="radio" name="obavezan" id="obavezan" value="1" checked>
<label class="form-check-label" for="flexRadioDefault2" value="1"> DA </label> 
<input class="form-check-input" type="radio" name="obavezan" id="obavezan" value="0" checked>
<label class="form-check-label" for="flexRadioDefault2"  value="0"> NE </label></br></br>
```

Slika 40. Prikaz koda za stupac obavezan tablice kolegij u create.blade.php

```
<label>OBAVEZAN </label>   

<input class="form-check-input" type="radio" name="obavezan" id="obavezan" value="1"
{{ old('obavezan', $kolegij->obavezan) == '1' ? 'checked' : '' }} >
<label class="form-check-label" for="flexRadioDefault2" value="1"> DA </label> 
<input class="form-check-input" type="radio" name="obavezan" id="obavezan" value="0"
{{ old('obavezan', $kolegij->obavezan) == '0' ? 'checked' : '' }} >
<label class="form-check-label" for="flexRadioDefault2"  value="0"> NE </label></br></br>
```

Slika 41. Prikaz koda za stupac obavezan tablice kolegij u edit.blade.php



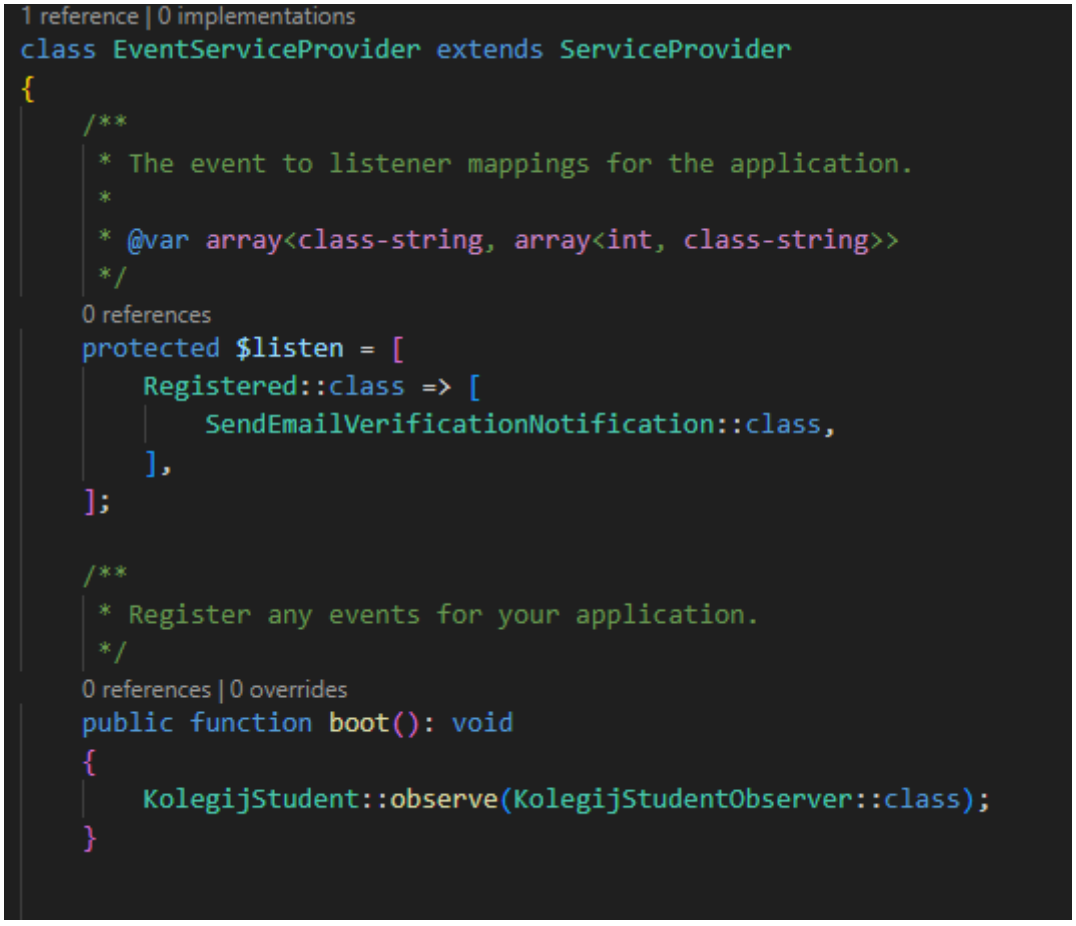
Slika 42. Prikaz forme za unos novog kolegija

Također, tablica student jedna od važnijih tablica u aplikaciji koja sadrži podatke o samom studentu koji se upisuje u višu godinu studija te ona sadrži stupce *JMBAG*, *OIB*, *Ime*, *Prezime*, *Spol*, *Datum_rodenja*, *Adresa_prebivalista*, *Adresa_boravista*, *Email*, *Mobitel*, *Telefon* i *Prosjek*. Upravo stupac *prosjek* je i najzanimljiviji pošto za njega je potrebno napraviti da se upisom položenih kolegija za studenta prosjek sam izračunava i ažurira. Kako bi se prosjek sam ažurirao potrebno je prvo napraviti novu klasu promatrač pomoću linije *php artisan make:observer StudentKolegij --model=StudentKolegij* koja služi za pokretanja metoda kada se stanje modela promijeni. U novog kreiranog promatrača potrebno je definirati metodu za računanje prosjeka na način da se pronađe student za kojeg je spremljen kolegij te nakon toga računa prosjek. Definiraju se dvije funkcije, prva je funkcija *saved()* koja služi kada se kolegij

spremi ili ažurira da se prosjek automatski ažurira te funkcija *deleted()* koja ima istu funkciju no pokreće se prilikom brisanja kolegija. Prosjek se računa tako da se iz klase *KolegijStudent* traže kolegiji koji pripadaju studentu tako da se usporedi *id_studenta_kolegij_student* s *id_studenta_kolegij_student* traženog studenta koji je spremljen u varijablu *$student*. Nakon čega, se traže kolegiji čija je ocjena veća 2 i provjerava se ukoliko kolegij ulazi u prosjek na način da se prvo traži *id_kolegija_kolegij_student* koji odgovara *id_kolegija* u tablici *kolegij* te u tablice *kolegij* se provjerava ukoliko *ulazi_u_prosjek* ima spremljenu vrijednost jednaku *1*. Na kraju se funkcijom *avg()* izračunava prosjek na temelju pronađenih kolegija koji zadovoljavaju sve zadane uvjete za studenta. Nakon što se izračuna prosjek i pohrani u varijablu *$prosjek* potrebno je u klasi *Student* pronaći studenta za kojeg se ažurira prosjek pomoću metodom *find()* te tom studentu ažurirati prosjek s metodom *update()*. Također, kako bi se promatrač povezao s ostalim kodom potrebno je u *Providers* pronaći datoteku *EventServiceProvide.php* i u njoj u funkciji *boot()* dodati kreiranog promatrača na način da se definira promatrač na klasu *KolegijStudent*. Slika koda za funkcije *saved()* i *delete()* prikazana je na Slici 43.

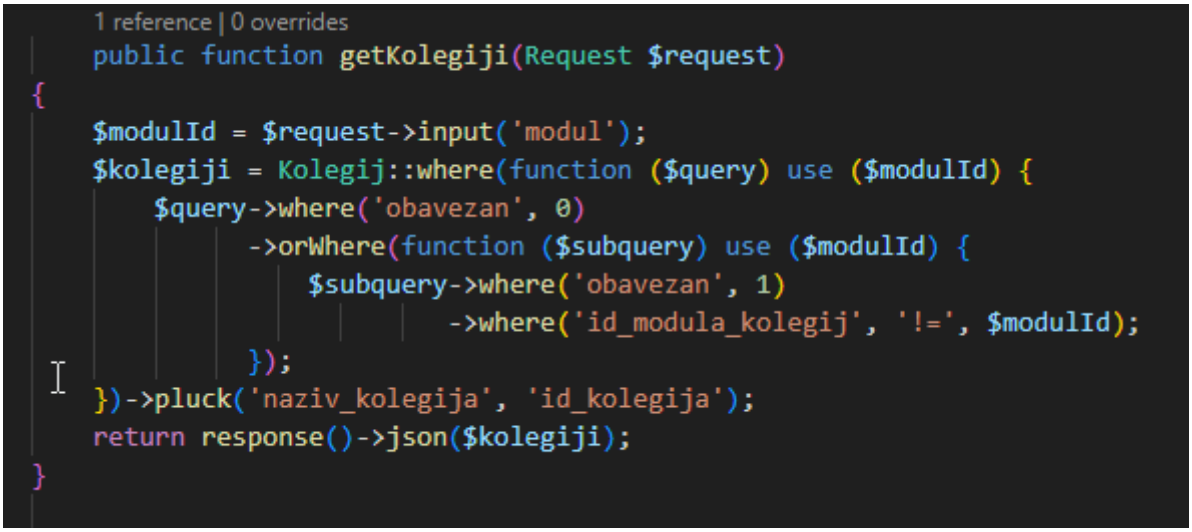
```php
<?php

namespace App\Observers;

use App\Models\KolegijStudent;
use App\Models\Student;

2 references | 0 implementations
class KolegijStudentObserver
{
    0 references | 0 overrides
    public function saved(KolegijStudent $kolegij_student)
    {

        $student = $kolegij_student->id_studenta_kolegij_student;
        $student1 = Student::find($student);

        $prosjek = KolegijStudent::where('id_studenta_kolegij_student', $student)->where('ocjena', '>=', 2)
        ->whereIn('id_kolegija_kolegij_student', function ($query) {
            $query->select('id_kolegija')->from('kolegij')->where('ulazi_u_prosjek', 1);
        })
        ->avg('ocjena');


        $student1->update(['prosjek' => $prosjek]);
    }
    0 references | 0 overrides
    public function deleted(KolegijStudent $kolegij_student)
{

    $student = $kolegij_student->id_studenta_kolegij_student;
    $student1 = Student::find($student);


    $prosjek = KolegijStudent::where('id_studenta_kolegij_student', $student)->where('ocjena', '>=', 2)
        ->whereIn('id_kolegija_kolegij_student', function ($query) {
            $query->select('id_kolegija')->from('kolegij')->where('ulazi_u_prosjek', 1);
        })
        ->avg('ocjena');


    $student1->update(['prosjek' => $prosjek]);
}
```

Slika 43. Prikaz KolegijStudentObserver.php datoteke



Slika 44. Prikaz EventServiceProvider.php datoteke

Tablica *upisi* je zapravo i najbitnija tablica pošto se pomoću nje vrše upisi u višu godinu studija. Proces kreiranja, modeliranja i uređivanja tablice je isti kao što je prethodno upisano za ostale tablice, dok za stupce za izbor izbornih kolegija postoje poseban način prikaza. Proces je da student prilikom upisa izabere kolegij na koji se upisuje te ovisno o njegovom odabiru za izborne kolegije prikazuju se kolegiji koji nisu obavezni na tom modulu. Prvo potrebno je definirati novu funkciju *getKolegiji()* u upravitellju *UpisiController.php* u kojoj se pronalaze kolegiji koji nisu obavezni za odabrani modul. Na način, da se pronalazi modul koji je odabran te sprema u varijablu *$modulId* nakon čega u varijablu *$kolegiji* se spremaju kolegiji. Kolegiji se pronalaze na način da se u klasi *Kolegij* traže kolegiji koji imaju isti i*d_modula_kolegij* kao i dohvaćeni modul za koje polje *obavezan* ima vrijednost *0* odnosno traže se izborni kolegiji odabranog modula. Također, s metodom or*Where()* pronalaze se i kolegiji koji imaju vrijednost polja *obavezan* na *1* ali *id_modula_kolegij* je različit od ID odabranog modula što znači da se uzimaju kolegiji koji su obavezni ali nisu obavezni na odabranom modulu. Nakon što su pronađeni svi kolegiji s metodom *pluck()* vraćaju se *naziv_kolegija* i *id_kolegija* te funkcija *return()* vraća listu dobivenih kolegija u JSON kako bi se moglo dalje konfigurirati.



```
1 reference | 0 overrides
public function getKolegiji(Request $request)
{
    $modulId = $request->input('modul');
    $kolegiji = Kolegij::where(function ($query) use ($modulId) {
        $query->where('obavezan', 0)
              ->orWhere(function ($subquery) use ($modulId) {
                  $subquery->where('obavezan', 1)
                           ->where('id_modula_kolegij', '!=', $modulId);
              });
    })->pluck('naziv_kolegija', 'id_kolegija');
    return response()->json($kolegiji);
}
```

Slika 45. Prikaz funkcije getKolegiji()

U *create.blade.php* potrebno je dodati javascript kako bi se definirao padajući izbornik na način da se prilikom svake promjene u polju modula ažurira lista kolegija. Za dodavanje javascripta potrebno je dodati link na javascript unutar sekcije <script> koji se dodaju na kraju dokumenta prije metode @stop. Te unutar druge sekcije <script> dodaje se kod koji poziva prethodno stvorenu metodu *getKolegiji()* i prikazuje kolegije unutar dinamičkog podajućeg izbornika. Kod započinje s linijom *$(document).ready(function()* koja služi da se kod unutar nje pokrene tek nakon što se cijeli dokument učita u web pregledniku. Potrebno je kada korisnik promijeni unos u polju modul da se zabilježi promjena pomoću linije *$('#id_modula_upisi').on('change', function()* koji povezuje *id_modula_upisi* sa slušateljem događanja. Nakon čega promjena se dohvaća i sprema u varijablu *selectedModul* u liniji *var selectedModul = $(this).val()* i pokreće se AJAX zahtjev prema linku *'/modul2'* u kojem se dohvaćaju kolegiji s metodom *GET* te linijom *data: { modul: selectedModul }* se šalje varijabla *selectedModul* s AJAX zahtjevom koja služi za identifikaciju modula. Također, da bi se dohvatili kolegiji unutar datoteke *web.php* potrebno je dodati rutu *'/modul2'* koja poziva funkciju *geKolegiji()* iz upravitelja *UpisiController.php* te cijela linija za dodavanje rute glasi *Route::get('/modul2', [UpisiController::class, 'getKolegiji'])->name('upisi.getKolegiji');*. Ako je AJAX zahtjev uspješan onda se sadržaj padajućih izbornika isprazni pomoću metode *empty()* tako da je svaki izbornik potrebno posebno isprazniti s linijom *$('#id_kolegija_zimski_prvi').empty();*. Isto tako, svakom padajućem izborniku se dodaje opcija „---“ koja se pokazuje prije odabira kolegija te opcija služi i za odabir na način da ukoliko korisnik nema izbornog kolegija koji bi odabrao postavlja „---“ kao njegov odabir.

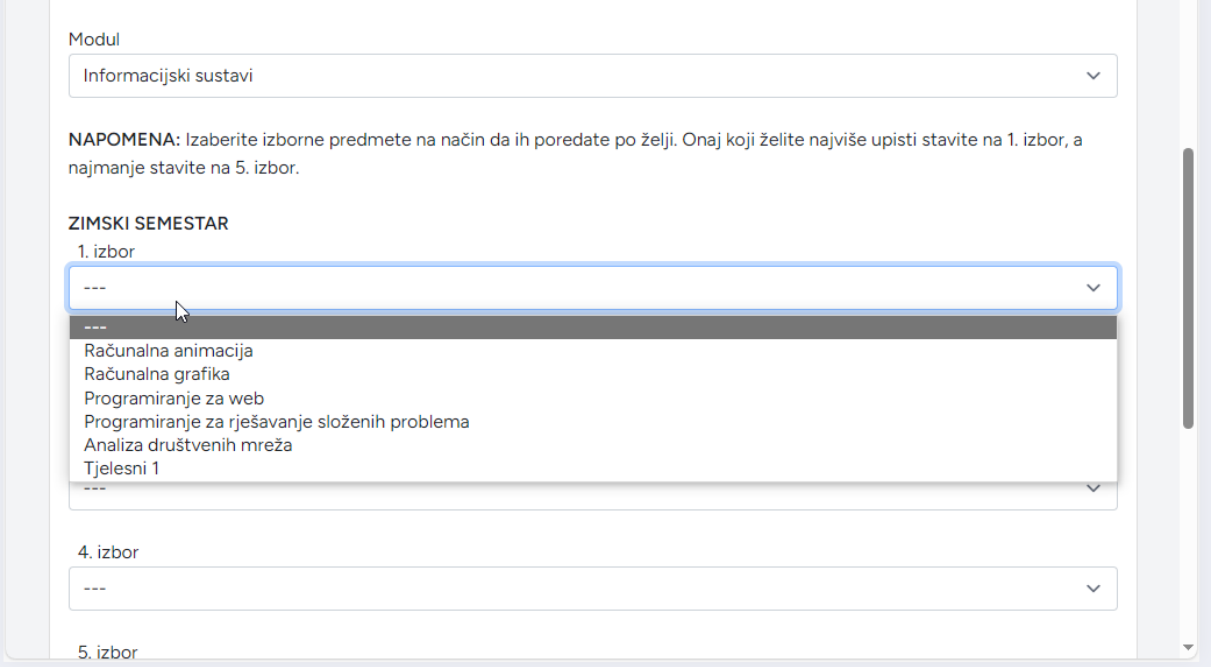
```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>


<script>
 $(document).ready(function() {
  $('#id_modula_upisi').on('change', function() {
    var selectedModul = $(this).val();
    $.ajax({
      url: '/modul2',
      method: 'GET',
      data: { modul: selectedModul },
      success: function(data) {

        $('#id_kolegija_zimski_prvi').empty();
        $('#id_kolegija_zimski_drugi').empty();
        $('#id_kolegija_zimski_treci').empty();
        $('#id_kolegija_zimski_cetvrti').empty();
        $('#id_kolegija_zimski_peti').empty();
        $('#id_kolegija_ljetni_prvi').empty();
        $('#id_kolegija_ljetni_drugi').empty();
        $('#id_kolegija_ljetni_treci').empty();
        $('#id_kolegija_ljetni_cetvrti').empty();
        $('#id_kolegija_ljetni_peti').empty();


        $('#id_kolegija_zimski_prvi').append('<option value="">---</option>');
        $('#id_kolegija_zimski_drugi').append('<option value="">---</option>');
        $('#id_kolegija_zimski_treci').append('<option value="">---</option>');
        $('#id_kolegija_zimski_cetvrti').append('<option value="">---</option>');
        $('#id_kolegija_zimski_peti').append('<option value="">---</option>');
        $('#id_kolegija_ljetni_prvi').append('<option value="">---</option>');
        $('#id_kolegija_ljetni_drugi').append('<option value="">---</option>');
        $('#id_kolegija_ljetni_treci').append('<option value="">---</option>');
        $('#id_kolegija_ljetni_cetvrti').append('<option value="">---</option>');
        $('#id_kolegija_ljetni_peti').append('<option value="">---</option>');
```

Slika 46. Prikaz javascripta unutar creta.blade.php datoteteke 1.dio

Nakon čega se puni dinamički padajući izbornik tako da se s metodom *each()* prolazi kroz svaki kolegij te za svaki kolegij u *value* pridružuje se varijabla *index* u kojem je spremljena vrijednost *id_kolegija_zimski_prvi* dok se na web pregledniku prikazuje naziv kolegija koji je spremljen u varijabli *kolegij*. Te s linijom *$('#id_modula_upisi').trigger('change')* se pokreće događaj promjene te popunjavaju se izbornici s kolegijima na temelju odabranog modula. Prikaz koda prikazana je na Slici 46. i Slici 47., dok prikaz web preglednika prilikom odabira modula je prikazan na Slici 48.

```
        $.each(data, function(index, kolegij) {
          $('#id_kolegija_zimski_prvi').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_zimski_drugi').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_zimski_treci').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_zimski_cetvrti').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_zimski_peti').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_ljetni_prvi').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_ljetni_drugi').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_ljetni_treci').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_ljetni_cetvrti').append('<option value="' + index + '">' + kolegij + '</option>');
          $('#id_kolegija_ljetni_peti').append('<option value="' + index + '">' + kolegij + '</option>');
        });
      }
    });
  });
  $('#id_modula_upisi').trigger('change');
});
</script>
```

Slika 47 Prikaz javascripta unutar creta.blade.php datoteteke 2.dio



Slika 48. Prikaz izbora izbornih predmeta

4. AUTORIZACIJA

Aplikacija je zamišljena da ima dva tipa korisnika admina i studenta koji bi se prijavili u aplikaciju i s obzirom na ulogu korisnika imali bi pristup različitim podacima. Kako bi se napravio login i autentifikacija korisnika u laravelu potrebno je koristit *larastrust* paket s laravel *breeze*. Nakon instalacije paketa pod *Https/Controlller* postoji nova mapa *Auth* koja sadrži sve potrebne upravitelje za login i registraciju korisnika. Također, dodatno su dodane tablice *roles*, *permisssions*, *role_user*, *permission_user* i *permission_role*. U tablicu *user* se dodaju novi korisnici te u tablicu *role* dodaju se uloge korisnika koje se povezuju s korisnikom preko tablice *role_user*. U aplikaciji potrebno je preusmjeriti prijavljenog korisnika na različite početne stranice s obzirom na ulogu koja je dodijeljena korisniku, te za kontrolu usmjerenja potrebno je kreirati novi upravitelj s nazivom *DashboardConttroller*. Nakon čega potrebno je definirati novu funkciju *indeks()* i unutar nje pomoću *if* naredbe provjeriti koji korisnik je prijavljen i vratiti odgovarajuću stranicu. Poziva se metoda *Auth::user()* kako bi se dohvatili podaci o prijavljenom korisniku te s funkcijom *hasRole ('admin')* provjerava se ako je njegova uloga admin odnosno ako je prijavljeni korisnik admin. Ukoliko je njegova uloga admin onda korisnika se preusmjerava na *main.blade.php* stranicu, a ako nije provjerava se naredba *elseif* ukoliko je prijavljeni korisnik student. Ako je prijavljeni korisnik student onda se korisnika preusmjerava na *mainstudent.blade.php* stranicu. Raspisani kod je prikazan na Slici 49.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

0 references | 0 implementations
class DashboardController extends Controller
{
    0 references | 0 overrides
    public function index()
    {
        if(Auth::user()->hasRole('admin')){
            return view('main');
        }
        elseif(Auth::user()->hasRole('student'))
        {
            return view('mainstudent');
        }
    }

    0 references | 0 overrides
    public function potvrda()
    {
     return view('potvrda');
    }
}

```

Slika 49. Prikaz koda u DashboardController datoteci

Datoteka *main.blade.php* nalazi se pod *views* unutar mape *resources* te u njoj se definira kako se prikazuje sadržaj na web stranici. U kodu u *<head>* sekciji uključen je *boostrap 5* oblikovanje stranice, dok u *<body>* sekciji koristi se *blade* komponenta *<x-app-layout>* koji je dio Laravela za definiranje izgleda web stranice. Unutar *blade* komponente s linijom *@if (request()->is('dashboard'))* provjera se ako se korisnik nalazi na ruti *'dashboard'* te ukoliko se nalazi onda se ispiše poruka dobrodošlice. Također, koristi se i *blade* direktiva *@yield('content')* koji služi za prijenos sadržaja s podređene stranice na glavnu stranicu. Što znači, ako se korisnik nalazi na ruti *'student'* prikaz će se glavna stranica definirana u *main.blade.php* datoteci i unutar komponente * * upisati će se sadržaj iz stranice *student.index* koji se dohvaća upravo pomoću *@yield('content')* komponente. Na isti način, uređuje se i *mainstudent.blade.php* datoteka čiji je kod prikaz na Slici 51., dok kod *main.blade.php* datoteke je prikazan na Slici 50.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE-edge">
    <meta name="viewport" cntent="withd-device-width, initial-scale-1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/
    <title>Upisi u više godine</title>
</head>

<body>
    <x-app-layout>
       <div class= "container-md">
        <div class="col">
            <div></br></div>
            @if (request()->is('dashboard'))
            <div class="max-w-7xl mx-auto p-6 lg:p-8">
                <div class="mt-16">
                    <div class="flex justify-center ">
                        <div class="scale-100 p-6 bg-white dark:bg-gray-800/50 dark:bg-gradient-to-bl from-gray-700/50 via-transpar

                            <h1>Dobrodošli u aplikaciju <b>Upisi u više  godine studija</b></br></br>
                            Ulogirani ste kao <b>{{Auth::user()->name}}</b>
                            <h1>
                        </div>
                    </div>
                </div>
            </div>
            @endif

                @yield('content')
        </div>
       <div>
    </x-app-layout>

</body>
</html>
```

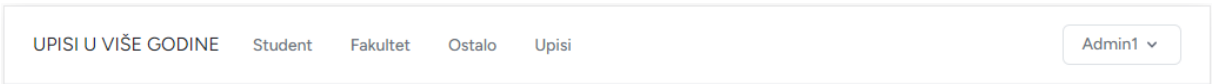
Slika 50. Prikaz koda u main.blade.php

```
</head>

<body>
<x-app-layout>
      <div class= "container">
       <div class="row">
       <div></br></div>
       @if (request()->is('dashboard'))
           <div class="mt-16">
               <div class="flex justify-center ">
                       <div class="scale-100 p-6 bg-white dark:bg-gray-800/50 dark:bg-gradient-to-bl from-gray-700/50 via-transparent dark:ring-1 dark:ring-inset

                           <h1>Dobrodošli u aplikaciju <b>Upisi u više  godine studija</b></br></br>
                           Ulogirani ste kao <b>{{Auth::user()->name}}</b>
                           </h1>

                           </div>
                   </div>
           </div>
               <div class="mt-16">
                   <div class="flex justify-center ">
                       <div class="scale-100 p-6 bg-white dark:bg-gray-800/50 dark:bg-gradient-to-bl from-gray-700/50 via-transparent dark:ring-1 dark:ring-inset

                           <h1>Upisi u više godine studija održavaju se od 10.09.2024 do 28.09.2024</h1></br>
                       </div>
                   </div>
           </div>
           <div class="mt-16">
                   <div class="flex justify-center ">

                           <a  href="{{ url('/student/'  ) }}" class="btn btn-success btn-sm" title="Započni upise"><i class="fa fa-plus" aria-hidden="true"></i>
                           Započni upise</a>
                   </div>
           </div>
       @endif
               @yield('content')

       </div>
    </div>
    </x-app-layout>
</body>

</html>
```

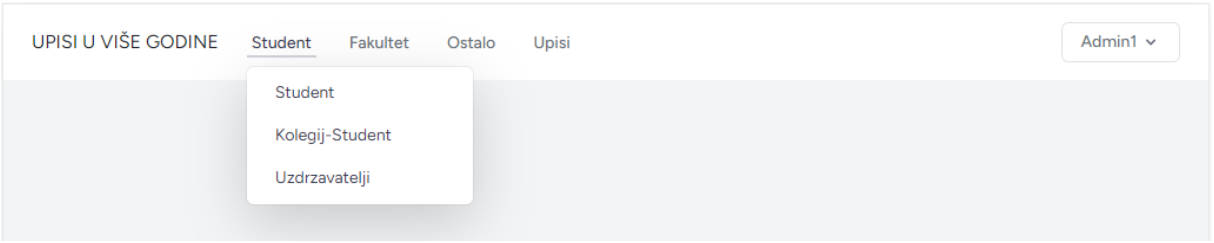
Slika 51. Prikaz koda u mainstudent.blade.php

4.1. NAVIGACIJA

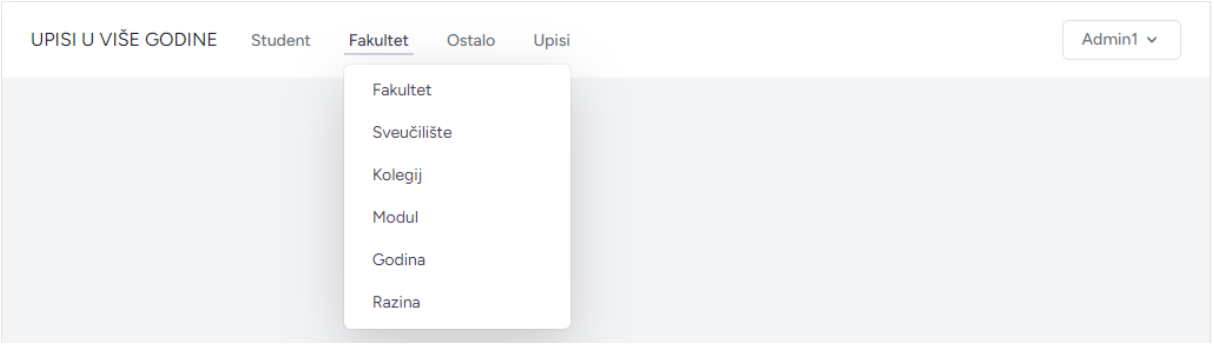
U prethodnom poglavlju raspravljeno je kako s obzirom na ulogu korisnika prikazuje se drugačije stranice pošto postoje dva tipa uloge prvi je *admin* koji ima pristup svim podacima i *student* koji ima pristup podacima povezanim samo za sebe. Navigacija je napravljena na način da ako je prijavljeni korisnik student da se prikazuju izbornici *Student*, *Uzdržavatelj*, *Upisi* i *Kolegij*, a ako je prijavljeni korisnik admin da se prikazuje padajući izbornici *Student*, *Fakultet*, *Ostalo* i izbornik *Upisi*. Padajući izbornik *Student* sadrži podatke vezane za studenta te sadrži elemente *Student*, *Student-Kolegij* i *Uzdržavatelj*, dok padajući izbornik *Fakultet* sadrži podatke o fakultetu te on sadrži elemente *Fakultet*, *Sveučilište*, *Kolegij*, *Modul*, *Godina*, i *Razina*. Padajući izbornik *Ostalo* sadrži ostale podatke u koje spadaju elementi *Država*, *Županija*, *Općina*, *Mjesto*, *Državljanstvo*, *Narodnost*, *Obrazovanje*, *Stručna sprema*, *Zanimanje*, *Položaj u zanimanju*.



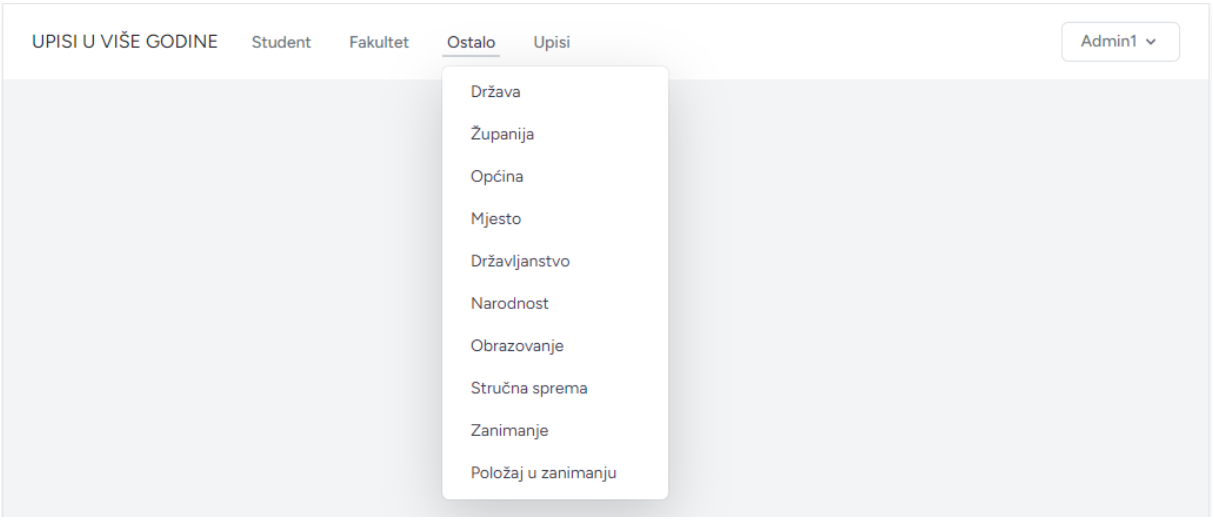
Slika 52. Prikaz izgleda navigacije za admina



Slika 53. Prikaz izgleda padajućeg izbornika Student za admina



Slika 54. Prikaz izgleda padajućeg izbornika Fakultet za admina



Slika 55. Prikaz izgleda padajućeg izbornika Ostalo za admina

Navigaciju se uređuje u datoteci *navigation.blade.php* koja se nalazi u *layouts* unutar mape *views* te ona je dodana prilikom instalacije breeze paketa. U kodu s linijom *@if(Auth::user()->hasRole('admin))* provjera se ako prijavljeni korisnik ima ulogu admina ukoliko da onda se prikazuje navigacija za admina. Unutar *@if* naredbe definira se za svaki izbornik njegovo oblikovanje i rutu na koju vodi link prilikom klika. Za izbornik *Student* definirano je da prilikom klika na njega da se otvori padajući izbornik te klikom na prvi element *Student* vodi na rutu '*student'* s podacima o studentima koji se dobivaju iz *student.index* datoteke. Dok drugi element iz padajućeg izbornika je *Kolegij-Student* koji vodi na rutu '*kolegijstudent'* na kojoj se prikazuju podaci o kolegijima koje je student položio te njihov prikaza definira se u *kolegijstudent.index* datoteci. Zadnji element je *Uzdržavatelj* te klikom na njega vodi na rutu '*uzdrzavatelj'* koja prikazuje podatke o uzdržavateljima koji se definiraju u datoteci *uzdrzavatelj.index*.

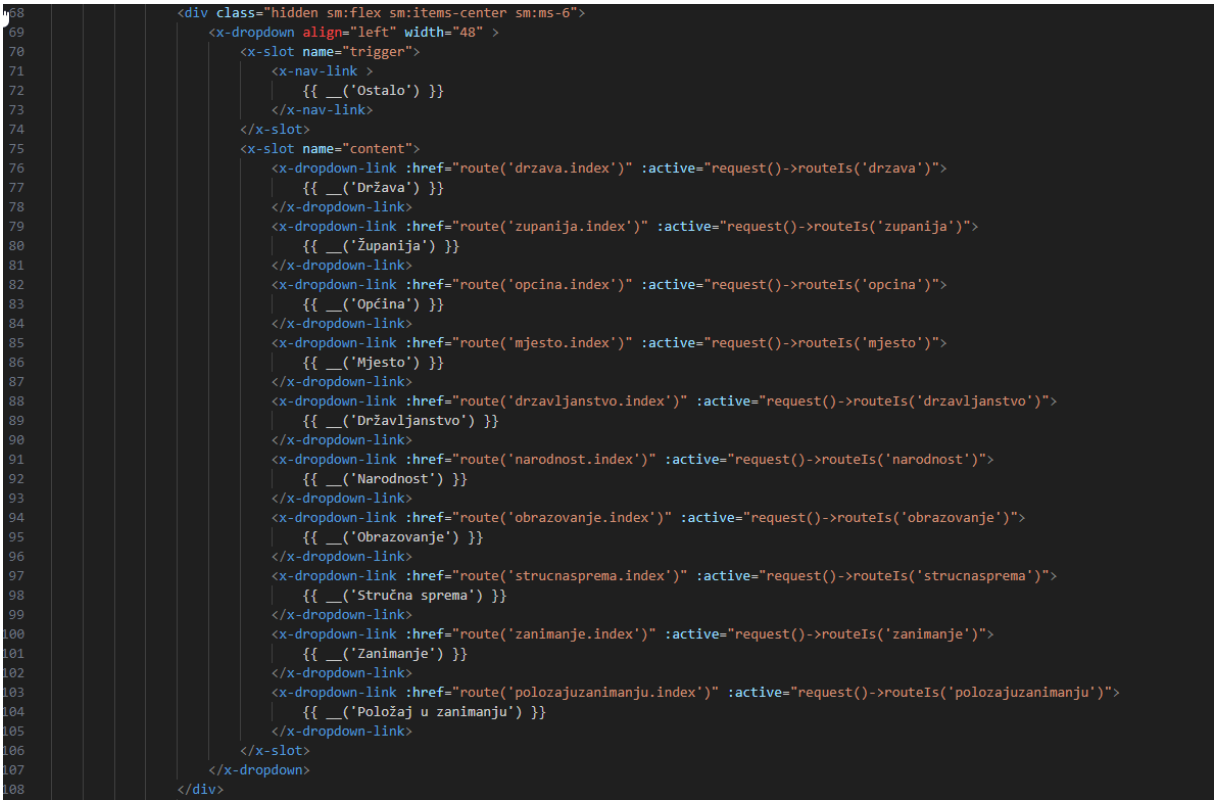
```
@if (Auth::user()->hasRole('admin'))
<div class="hidden sm:flex sm:items-center sm:ms-6">
    <x-dropdown align="left" width="48"  >
        <x-slot name="trigger">
            <x-nav-link >
                {{ __('Student') }}
            </x-nav-link>
        </x-slot>
        <x-slot name="content">
        <x-dropdown-link :href="route('student.index')" :active="request()->routeIs('student')">
                {{ __('Student') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('kolegijstudent.index')" :active="request()->routeIs('kolegijstudent')">
                {{ __('Kolegij-Student') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('uzdrzavatelj.index')" :active="request()->routeIs('uzdrzavatelj')">
                {{ __('Uzdrzavatelji') }}
            </x-dropdown-link>
        </x-slot>
    </x-dropdown>
</div>
```

Slika 56. Prikaz koda za padajući izbornika Student za admina

Po principu opisanom za izbornik *Student* rade se i padajući izbornici *Fakultet*, *Ostalo* i izbornik *Upisi*, na način da se definira ruta na koju se vodi nakon klika na element i datoteka iz koje se definira prikaz podataka. Kod za *Fakultet* je prikazan na Slici 57., kod *Ostalo* na Slici 58. i kod za *Upise* na Slici 59. Te nakon završetka pisanje navigacije u *@if* naredbu potrebno je zatvoriti sa *@endif* naredbom.

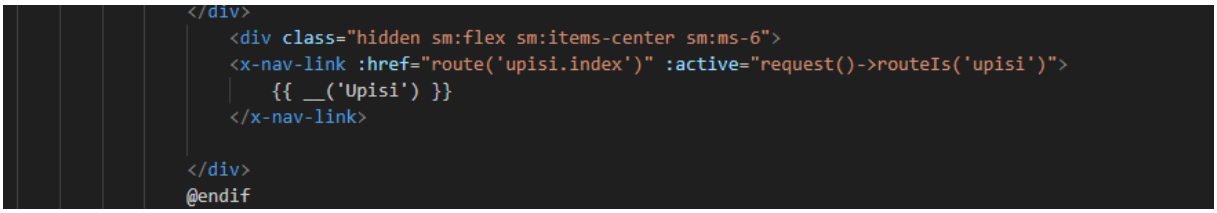
```
<div class="hidden sm:flex sm:items-center sm:ms-6">
    <x-dropdown align="left" width="48"  >
        <x-slot name="trigger">
            <x-nav-link >
                {{ __('Fakultet') }}
            </x-nav-link>
        </x-slot>
        <x-slot name="content">
        <x-dropdown-link :href="route('fakultet.index')" :active="request()->routeIs('fakultet')">
                {{ __('Fakultet') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('sveuciliste.index')" :active="request()->routeIs('sveuciliste')">
                {{ __('Sveučilište') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('kolegij.index')" :active="request()->routeIs('kolegij')">
                {{ __('Kolegij') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('modul.index')" :active="request()->routeIs('modul')">
                {{ __('Modul') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('godina.index')" :active="request()->routeIs('godina')">
                {{ __('Godina') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('razina.index')" :active="request()->routeIs('razina')">
                {{ __('Razina') }}
            </x-dropdown-link>
        </x-slot>
    </x-dropdown>
</div>
```

Slika 57. Prikaz koda za padajući izbornika Fakultet za admina



```
<div class="hidden sm:flex sm:items-center sm:ms-6">
    <x-dropdown align="left" width="48" >
        <x-slot name="trigger">
            <x-nav-link >
                {{ __('Ostalo') }}
            </x-nav-link>
        </x-slot>
        <x-slot name="content">
            <x-dropdown-link :href="route('drzava.index')" :active="request()->routeIs('drzava')">
                {{ __('Država') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('zupanija.index')" :active="request()->routeIs('zupanija')">
                {{ __('Županija') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('opcina.index')" :active="request()->routeIs('opcina')">
                {{ __('Općina') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('mjesto.index')" :active="request()->routeIs('mjesto')">
                {{ __('Mjesto') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('drzavljanstvo.index')" :active="request()->routeIs('drzavljanstvo')">
                {{ __('Državljanstvo') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('narodnost.index')" :active="request()->routeIs('narodnost')">
                {{ __('Narodnost') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('obrazovanje.index')" :active="request()->routeIs('obrazovanje')">
                {{ __('Obrazovanje') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('strucnasprema.index')" :active="request()->routeIs('strucnasprema')">
                {{ __('Stručna sprema') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('zanimanje.index')" :active="request()->routeIs('zanimanje')">
                {{ __('Zanimanje') }}
            </x-dropdown-link>
            <x-dropdown-link :href="route('polozajuzanimanju.index')" :active="request()->routeIs('polozajuzanimanju')">
                {{ __('Položaj u zanimanju') }}
            </x-dropdown-link>
        </x-slot>
    </x-dropdown>
</div>
```

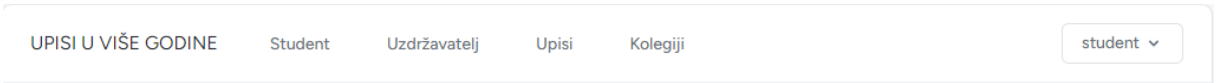
Slika 58. Prikaz koda za padajući izbornika Ostalo za admina



```
</div>
    <div class="hidden sm:flex sm:items-center sm:ms-6">
    <x-nav-link :href="route('upisi.index')" :active="request()->routeIs('upisi')">
        {{ __('Upisi') }}
    </x-nav-link>

</div>
@endif
```

Slika 59. Prikaz koda za izbornik Upisi za admina

Također, koristeći prethodno opisani princip radi se provjera ukoliko prijavljeni korisnik je student pomoću linije *@if(Auth::user()->hasRole('student'))*. Ukoliko korisnik ima ulogu studenta potrebno je prikazati izbornike *Student*, *Uzdržavatelj*, *Upisi* i *Kolegij* kao što je prikazano na Slici 60. Unutar naredbe *@if* za svaki izbornik potrebno je definirati rutu na koju vodi link nakon klika na element i datoteku iz koje se definira prikaz podataka. S time da kod studenta postoji izbornik *Kolegij* koji vodi na link '*kolegijstudent*' i uzima podatke iz tablice *kolegij_student* čiji je prikaz definiran u datoteci *kolegijstudent.index*. Razlog tome je da se želi studentu prikazati popis svih kolegija koje je položio. Te nakon završetka pisanje navigacije potrebno je zatvoriti *@if* s *@endif*. Cijeli kod za definiranje navigacije za studenta je prikazan na Slici 61.



Slika 60. Prikaz izgleda navigacije za studenta

```
@if (Auth::user()->hasRole('student'))
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
    <x-nav-link :href="route('student.index')" :active="request()->routeIs('student')">
        {{ __('Student') }}
    </x-nav-link>
</div>
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
    <x-nav-link :href="route('uzdrzavatelj.index')" :active="request()->routeIs('uzdrzavatelj')">
        {{ __('Uzdržavatelj') }}
    </x-nav-link>
</div>
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
    <x-nav-link :href="route('upisi.index')" :active="request()->routeIs('upisi')">
        {{ __('Upisi') }}
    </x-nav-link>
</div>
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
    <x-nav-link :href="route('kolegijstudent.index')" :active="request()->routeIs('kolegijstudent')">
        {{ __('Kolegiji') }}
    </x-nav-link>
</div>
@endif
</div>
```

Slika 61. Prikaz koda za navigaciju za studena

Osim preko navigacije do podataka moguće je pristupiti i upisom direktnog linka na web pregledniku čime se dolazi do situacije da student ili ne prijavljeni korisnik može pristupiti svim tablicama. No, kako student ne smije imati pristup svim tablicama potrebno mu je ograničiti pristup samo na tablice *Student*, *Uzdržavatelj*, *Upisi* i *Kolegij*. Pristup tablicama ograničujemo u datoteci *web.php* koja se nalazi u datoteci *routes* te u njoj se definiraju sve rute koje su potrebne za kretanje po web stranici i pravila pristupa tim rutama s obzirom na ulogu koju korisnik ima. Ruti '/' nije postavljeno ograničenje i za pristupu toj rutu nije potrebna autentifikacija te ona funkcijom *return view ('welcome')* vraća *welcome.blade.php* stranicu (Slika 62?). Dok za ostale rute potrebno je postaviti ograničenja za pristup koja se postavljaju s obzirom na ulogu korisnika.

S obzirom na navedeno, postoje tri slučaja ograničenja pristupa u aplikaciji, prvi je davanje pristupa određenim rutama svim korisnicima koji su prijavljeni bez obzira na njihovu uloga. U ovom slučaju, admin i student imaju pristup rutama *'/student'*, *'/upisi'*, *'/kolegijstudent'* i *''/uzdrzavatelj'*. U kodu ograničenje pristupa je napravljeno pomoću funkcije *Route:group* koja služi za grupiranje ruta *te (['middleware' => ['auth']], function(){})* provjera se ako je korisnik prijavljen te nakon logina daje pristup upisanim rutama. S linijom *Route:resource ('/student', StudentController::class)* definira se naziv rute te što se događa ako se dođe do te rute na webu. U primjeru definirano je da se za rutu *'/student'* poziva upravitelj *StudentController* kao klasa te s obzirom na to kako su u njoj definirane funkcije tako se prikazuju i podaci na webu. Na isti način napravljeno je i za rute *'/upisi'*, *'/kolegijstudent'* i *''/uzdrzavatelj'* (Slika 62).

```php
Route::get('/', function () {
    return view('welcome');
});

Route::group(['middleware' => ['auth']], function() {
    Route::get('/dashboard', 'App\Http\Controllers\DashboardController@index')->name('dashboard');
    Route::resource('/student',StudentController::class)->middleware('auth');
    Route::resource('/upisi',UpisiController::class);
    Route::resource('/kolegijstudent',KolegijStudentController::class);
    Route::resource('/uzdrzavatelj',UzdrzavateljController::class);

});
```

Slika 62. Prikaz koda za usmjeravanje i pristup podataka 1.dio

Sljedeći slučaj, je davanje pristupa rutama samo korisnicima koji su prijavljeni i imaju ulogu admina što znači da student nema pristup njima. Ograničenje pristupa se radi na način da se provjera ako je korisnik prijavljeni te nakon logina koja je njegova uloga. Ako je njegova uloga admin onda ima pristup navedenim rutama. Provjera se radi s istom funkcijom objašnjenom u prethodnom odlomku, no za razliku od prije u uvjetu funkcije potrebno je dodati *'role:admin'* pa cijela funkcija bi glasila *Route:group(['middleware' => ['auth', 'role:admin']], function(){}).* Rute kojima admin ima pristup, a student nema jesu*: '/drzava', '/drzavljanstvo', '/narodnost', '/sveuciliste', '/obrazovanje', 'zanimanje', '/polozajuzanimanju', '/strucnasprema', '/razina', '/zupanija', '/opcina', '/mjesto', '/fakultet', '/godina', '/kolegij', '/modul'*. Za svaku od navedenih ruta potrebno je pozvati funkciju *Route::resource()* te u funkciju upisati naziv rute i dodati upravitelja kako bi se dolaskom na rutu prikazali željeni podaci. Primjer svih definiranih ruta i postupak ograničenja pristupa prikazano je na Slici 63.

```php
Route::group(['middleware' => ['auth', 'role:admin']], function() {

    Route::resource('/drzava',DrzavaController::class);
    Route::resource('/drzavljanstvo',DrzavljanstvoController::class);
    Route::resource('/narodnost',NarodnostController::class);
    Route::resource('/sveuciliste',SveucilisteController::class);
    Route::resource('/obrazovanje',ObrazovanjeController::class);
    Route::resource('/zanimanje',ZanimanjeController::class);
    Route::resource('/polozajuzanimanju',PolozajUZanimanjuController::class);
    Route::resource('/strucnasprema',StrucnaSpremaController::class);
    Route::resource('/razina',RazinaController::class);
    Route::resource('/zupanija',ZupanijaController::class);
    Route::resource('/opcina',OpcinaController::class);
    Route::resource('/mjesto',MjestoController::class);
    Route::resource('/fakultet',FakultetController::class);
    Route::resource('/godina',GodinaController::class);
    Route::resource('/kolegij',KolegijController::class);
    Route::resource('/modul',ModulController::class);
});
```

Slika 63. Prikaz koda za usmjeravanje i pristup podataka 2.dio

Posljednje ograničenje pristupa koje se postavlja je davanje pristupa rutama samo korisnicima koji su prijavljeni i imaju ulogu studenta. U ovom slučaju, je zapravo pristup ruti '/potvrda' na kojeg se preusmjeri studenta nakon što klikne na dugme *„Pošalji upise“*. Princip ograničenja pristupa je isti kao i za prethodne slučaj samo što se u funkciji kod provjere za uvjet postavlja *['middleware' => ['auth', 'role:student']]* kako bi se dalo pristup samo korisnicima koji imaju ulogu studenta. Kako za usmjerenje na rutu nema posebnog upravitelja koji je samo za nju zbog toga što se na toj ruti ne šalju podaci iz tablice pa je i način definiranje rute nešto drugačiji. Kao i prethodno koristi se funkcija *Route()* ali uz funkciju *get()* te u uvjet postavlja se naziv rute, a na mjestu upravitelja postavlja se putanja za *DashboardController* s time da na kraju putanje se dodaje *@potvrda* kako bi se pozvala samo funkcija *potvrda()* koja je definira u upravitelju.

```
Route::group(['middleware' => ['auth', 'role:student']], function() {
    Route::get('/potvrda', 'App\Http\Controllers\DashboardController@potvrda')->name('potvrda');

});
```

Slika 64. Prikaz koda za usmjeravanje i pristup podataka 3.dio

Također, kako bi se mogle koristiti sve potrebne funkcije i datoteke potrebno je u datoteku *web.php* navesti sve putanje korištenih funkcija i datoteka. Sve putanje prikazane su na Slici 65.

```
use App\Http\Controllers\KolegijStudentController;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\StudentController;
use App\Http\Controllers\DrzavljanstvoController;
use App\Http\Controllers\NarodnostController;
use App\Http\Controllers\SveucilisteController;
use App\Http\Controllers\ObrazovanjeController;
use App\Http\Controllers\ZanimanjeController;
use App\Http\Controllers\PolozajUZanimanjuController;
use App\Http\Controllers\StrucnaSpremaController;
use App\Http\Controllers\RazinaController;
use App\Http\Controllers\ZupanijaController;
use App\Http\Controllers\OpcinaController;
use App\Http\Controllers\MjestoController;
use App\Http\Controllers\FakultetController;
use App\Http\Controllers\UzdrzavateljController;
use App\Http\Controllers\UpisiController;
use App\Http\Controllers\GodinaController;
use App\Http\Controllers\KolegijController;
use App\Http\Controllers\ModulController;
use App\Http\Controllers\DrzavaController;
```

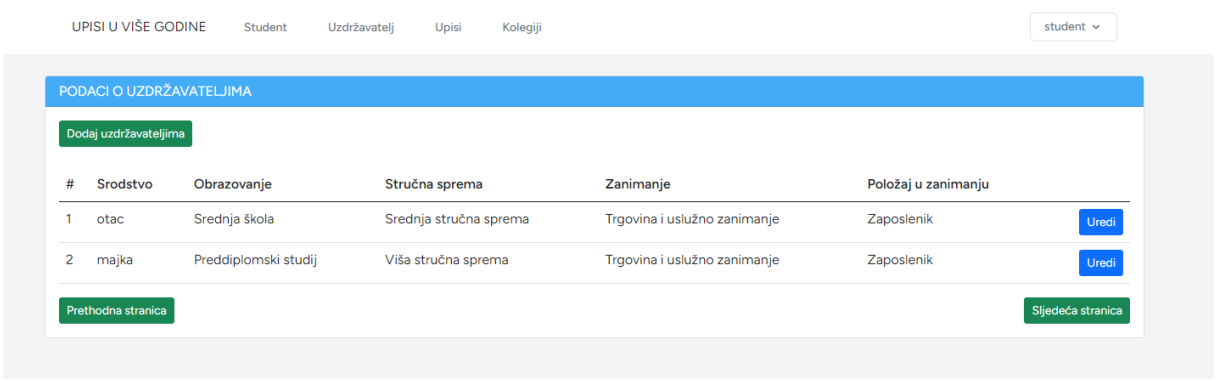
Slika 65. Prikaz uključenih putanja u web.php datoteci

5. RAZLIČITI PRIKAZ TABLICA STUDENT, UPISI, UZDRZAVATELJ I KOLEGIJ_STUDENT S OBZIROM NA ULOGU KORISNIKA

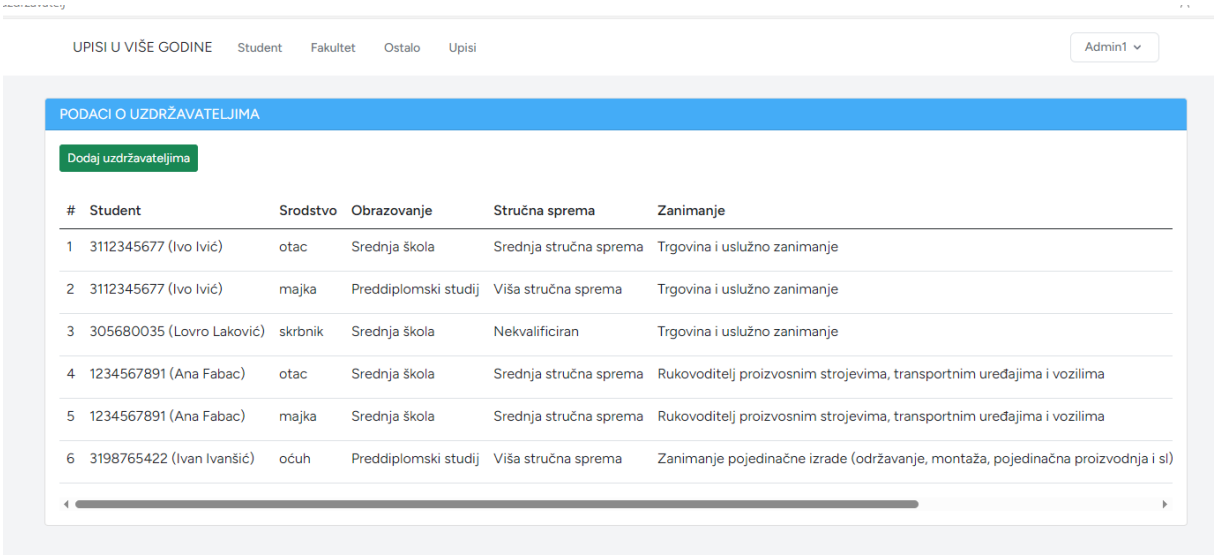
U aplikaciji upisi u više godine potrebno je s obzirom na login ograničiti i podatke u tablicama *student, kolegij_student, upisi* i *uzdrzavatelj*. Što se tiče tablice uzdržavatelj potrebno je studentu prikazati samo njegove uzdržavatelje dok admin ima prikaz svih uzdržavatelja. Također, student ne može brisati i prikazati uzdržavatelja kao i dodavanje studenta uzdržavatelju dok admin ima pristup svim navedenim opcijama. Ograničavanje u kodu započinje u upravitelju *UzdrzavateljController.php* tako da u funkciji *index()* se dodaje provjera o ulozi korisnika koji je prijavljen. Provjera se radi uz pomoć naredbe *if* i metode *Auth::user()* kako bi se dohvatili podaci o prijavljenom korisniku te s funkcijom *hasRole ('admin')* provjerilo ako je njegova uloga admin. Ukoliko je uloga korisnika admin onda se u varijablu *$uzdrzavatelj* spremaju svi podaci o svim uzdržavateljima te s metodama *return() i view()* vraća se *'uzdrzavatelj.index' te* uz pomoću metode *with()* šalju se podaci koji su spremljeni u varijablu *$uzdrzavatelj*. No ukoliko uloga korisnika nije admin onda se pomoću *elseif* naredbe opet ulazi u provjeru s uvjetom *Auth::user() -> hasRole ('admin')* čime se provjerava ako je uloga prijavljenog korisnika ipak student. Ukoliko je uloga korisnika student potrebno je pronaći o kojem studentu se radi te ispisati podatke samo za njega. Što znači, da se u klasi Student traži student pomoću metode *where()* kojoj je potrebno proslijediti uvjet *'email', '=', $usermail* kojim provjerava ako e-mail adresa korisnika odgovara student te s metodom *get()* pronađeni student se sprema u varijablu *$studentonly*. Nakon čega se isti postupak ponavlja i na klasi *Uzdrzavatelj* u kojoj se traže svi uzdržavatelji koji pripadaju pronađenom studentu tako da se provjerava ako *id_studenta_uzdrzavatelj* odgovara *JMBAG* pronađenog studenta. Svi podaci o pronađenim uzdržavateljima se spremaju u varijablu *$uzdrzavatelj* te sa metodama *return() i view()* vraća se *'uzdrzavatelj.index'* ali ovoga puta samo uz podatke za prijavljenog studenta. Cijeli kod za funkciju indeks je prikana na Slici 66., dok web prikaz tablice za studenta je prikazan na Slici 67., a admina na Slici 68.

```php
2 references | 0 implementations
class UzdrzavateljController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    0 references | 0 overrides
    public function index()
    {
        if(Auth::user()->hasRole('admin'))
        {
        $uzdrzavatelj=Uzdrzavatelj::all();
        return view('uzdrzavatelj.index')->with('uzdrzavatelj',$uzdrzavatelj);
        }
        elseif(Auth::user()->hasRole('student'))
        {
        $usermail=Auth::user()->email;
        $studentonly=Student::where('email', '=', $usermail)->get()->first();
        $uzdrzavatelj=Uzdrzavatelj::where('id_studenta_uzdrzavatelj', '=', $studentonly->JMBAG)->get();
        return view('uzdrzavatelj.index')->with('uzdrzavatelj',$uzdrzavatelj);
    }


    }
```

Slika 66. Prikaz index() funkcije u UzdrzavateljController.php



Slika 67. Prikaz popisa uzdržavatelja kada je korisnik prijavljen kao student



Slika 68. Prikaz popisa uzdržavatelja kada je korisnik prijavljen kao admin

Kako student može dodati uzdržavatelja potrebno je i u funkciji *create()* ograničiti da student može dodati uzdržavatelja samo za sebe. Kao i prethodno, provjerava se koja je uloga korisnika te ako je admin u varijabli *$student* se spremaju svi *JMBAG*, no ako je uloga korisnika student onda se varijabli spremaju samo *JMBAG* od studenta koji je prijavljen. Cijeli kod za *create()* prikazan je na Slici 69. S time se postiglo da kod kreiranja novog uzdržavatelja student ne može vidjeti podatke od ostalih studenta te u datoteci *create.balde.php* je potrebno ograničiti da se polje za stupac *id_studenta_uzdrzavatelj* samo popunjava ovisno o prijavljenom studentu.

```
0 references | 0 overrides
public function create()
{
    if(Auth::user()->hasRole('admin'))
    {
        $student=Student::pluck('JMBAG');
    }
    elseif(Auth::user()->hasRole('student'))
    {
        $usermail=Auth::user()->email;
        $studentonly=Student::where('email', '=', $usermail)->get()->first();
        $student=[$studentonly->JMBAG];

    }

    $srodstvo=['otac','majka','sestra','brat', 'maćuha','očuh','skrbnik'];
    $obrazovanje=Obrazovanje::pluck('naziv_obrazovanja','id_obrazovanja');
    $strucna_sprema=StrucnaSprema::pluck('naziv_strucne_spreme','id_strucne_spreme','kratica');
    $zanimanje=Zanimanje::pluck('naziv_zanimanja','id_zanimanja');
    $polozaj_u_zanimanju=PolozajUZanimanju::pluck('naziv_polozaja_u_zanimanju','id_polozaja_u_zanimanju');
    return view('uzdrzavatelj.create',compact('srodstvo','student','obrazovanje','strucna_sprema','zanimanje','polozaj_u_zanimanju'));
}
```

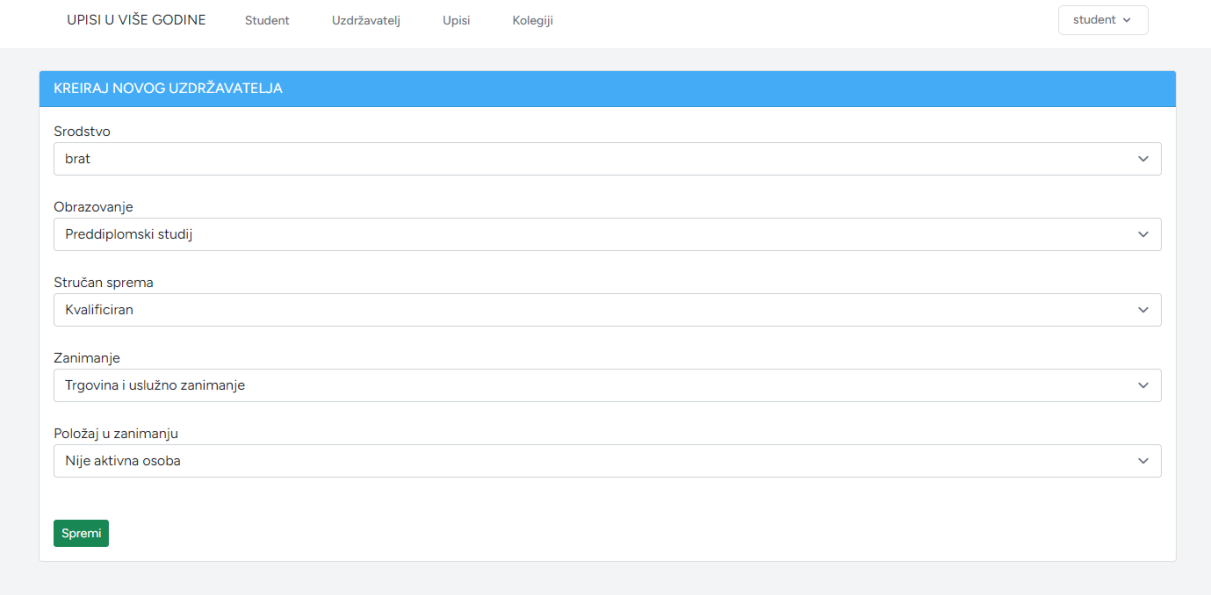
Slika 69. Prikaz create() funkcije u UzdrzavateljController.php

Za uređivanja polja *id_studenta_uzdrzavatelj* u datoteci *creat.blade.php* prvo je potrebno provjeriti koja je uloga korisnika te ako je uloga korisnika admin onda je potrebno prikazati sve studente u padajućem izborniku. No, ako je uloga korisnika student onda je potrebno sakriti polje od studenta na način da se *type* postavi na *hidden* za input polje te potrebno je i proslijediti podatke iz varijable *$student* kako bi se ispunio stupac *id_studenta_uzdrzavatelj* s podacima o studentu koji je prijavljen. Kod je prikazan na Slici 70. dok prikaz za forme za kreiranje

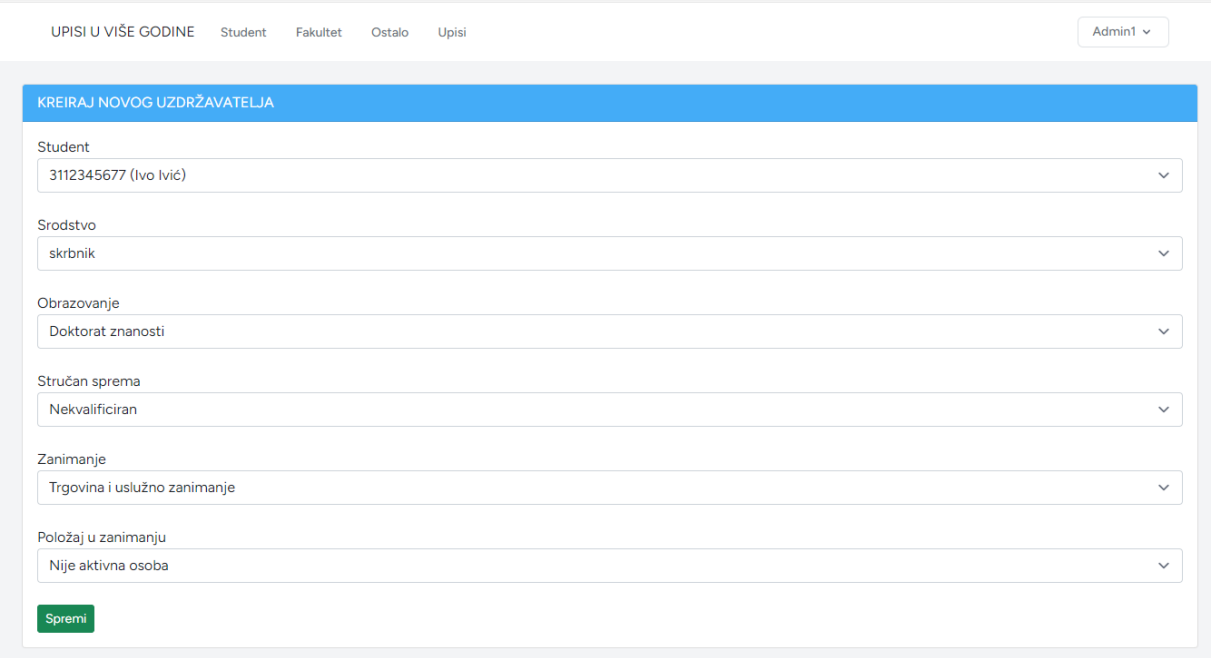
uzdržavatelja kada korisnik ima ulogu studenta prikaz je na Slici 65., a kada korisnik ima ulogu admina je prikazan na Slici 71.

```
@if(Auth::user()->hasRole('admin'))
<label>Student</label></br>
<select name="id_studenta_uzdrzavatelj" id="id_studenta_uzdrzavatelj" class="form-control">
  @foreach($student as $id)
    <option value="{{ $id}}">{{ $id}} </option>
  @endforeach
</select></br>
@elseif (Auth::user()->hasRole('student'))
@foreach($student as $id)
  <input  type="hidden" name="id_studenta_uzdrzavatelj" id="id_studenta_uzdrzavatelj" value="{{$id}}" class="form-control"></br>
  @endforeach
@endif
```

Slika 70. Prikaz provjere za stupac id_studenta_uzdrzavatelj *u datoteci* creat.blade.php



Slika 71. Prikaz forme za kreiranje uzdržavatelja kada je korisnik ima ulogu studenta

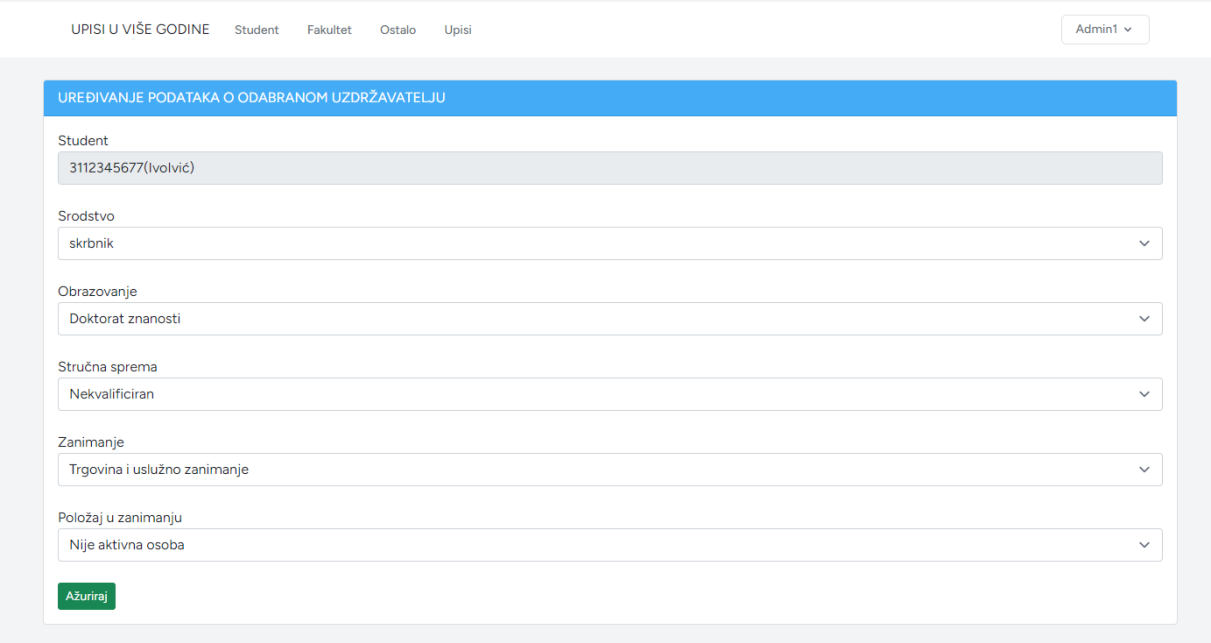


Slika 72. Prikaz forme za kreiranje uzdržavatelja kada je korisnik ima ulogu admina

Također, kod forme za uređivanje podataka o uzdržavatelju potrebno je kod admina onemogućiti uređivanje polja *id_studenta_uzdrzavatelj* pošto je uzdržavatelj povezan s studentom na način da jedan uzdržavatelj pripada tom studentu odnosno on je dio primarnog ključa tablice pa njegova promjena se ne smije dogoditi. U kodu to je potrebno riješiti na način da na kraju definiranja input polja potrebno dodati *disabled* čime se polje na webu onemogući uređivanje, no u polju se prikazuje izbor odabran prilikom kreiranja. Kod je prikaz na Slici 73., dok prikaz na webu je na Slici 74.

```
@if(Auth::user()->hasRole('admin'))
<label>Student</label></br>
<input type="string" name="id_studenta_uzdrzavatelj" id="id_studenta_uzdrzavatelj" value="{{$uzdrzavatelj->id_studenta_uzdrzavatelj }}"
class="form-control"  disabled  ></br>
@elseif (Auth::user()->hasRole('student'))
<input type="hidden" name="id_studenta_uzdrzavatelj" id="id_studenta_uzdrzavatelj" value="{{$uzdrzavatelj->id_studenta_uzdrzavatelj }}"
@endif
```

Slika 73. Prikaz koda za onemogućivanje uređivanja polja id_studenta_uzdrzavatelj



Slika 74 Prikaz forme za uređivanje uzdržavatelja kada je korisnik ima ulogu admina

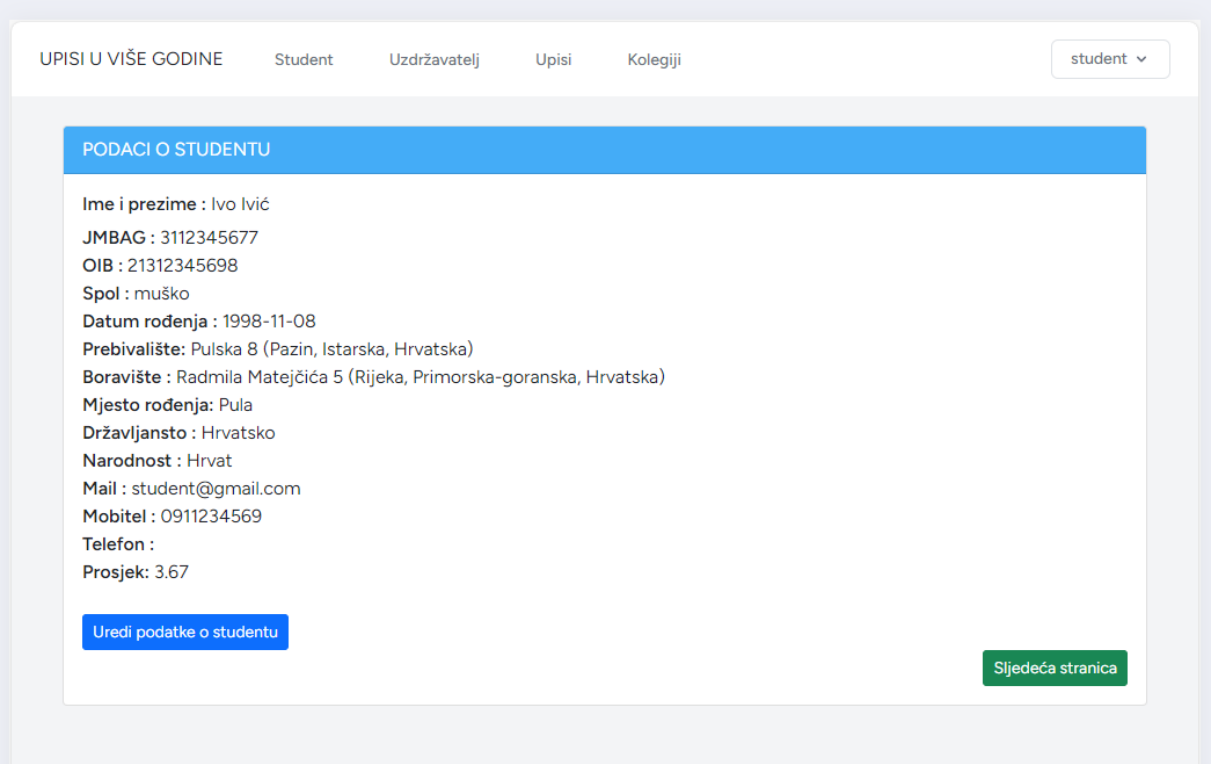
Što se tiče tablice *student, kolegij_student* i *upisa* ograničenje se radi na isti način kao i kod tablice *uzdrzavatelj*. Za tablicu *student* potrebno je ograničiti prikaz podataka na način da ako korisnik ima ulogu studenta da se prikazuju podaci samo o njemu. Također, za stupce *JMBAG* i *email* potrebno je onemogućiti uređivanje pošto njih definira administrator te oni služe za identifikaciju korisnika. Odnosno stupac JMBAG je primarni ključ tablice dok stupac *email* služi za identifikaciju koja je uloga korisnika i za login u aplikaciju. Prikaz podataka o studentu prikazuje kao što je definirano u *index.blade.php* kada je prijavljen student pošto se prikazuju podaci samo o njemu. Preusmjeravanje na datoteku *index.blade.php* se definira u upravitelju u funkciji *index()* na način da se provjeri uloga prijavljenog korisnika te ako je student vraćaju se podaci samo o prijavljenom studentu i preusmjeri na datoteku koja se nalazi u mapi *studentonly*. Ukoliko je uloga prijavljenog korisnika admin onda se vraćaju podaci o svim studentima i preusmjeri se na *index.blade.php* datoteku u mapi student. Također, ukoliko je prijavljen student prikazuje se još dva gumba koji služe za uređivanje podataka o studentu i za prelazak na sljedeću stranicu. Prikaz podataka u web pregledniku je prikazana na Slici 77.,Slici 78. i Slici 79. dok kod je prikazan na Slici 75.i Slici 76.

```php
class StudentController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    0 references | 0 overrides
    public function index()
    {
        if(Auth::user()->hasRole('admin')){
            $student=Student::all();
            return view('student.index')->with('student',$student);
        }
        elseif(Auth::user()->hasRole('student'))
            {
            $usermail=Auth::user()->email;
            $studentonly=Student::where('email', '=', $usermail)->get()->first();
            return view('studentonly.index')->with('studentonly',$studentonly);
        }

    }
```

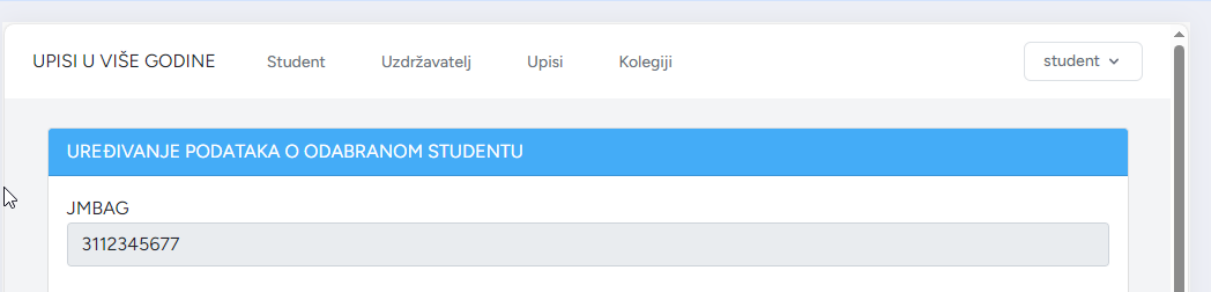
Slika 75. Prikaz index() funkcije u StudentController datoteci

```blade
@extends('main')


@section('content')


<div class="card">
  <div class="card-header text-white" style="background-color: #44ACF7;">PODACI O STUDENTU</div>
  <div class="card-body">

        <h5 class="card-title"><b>Ime i prezime :</b> {{ $studentonly->ime }} {{ $studentonly->prezime }}</h5>
        <p class="card-text"><b>JMBAG : </b>{{ $studentonly->JMBAG }}</p>
        <p class="card-text"><b>OIB : </b> {{ $studentonly->OIB }}</p>
        <p class="card-text"><b>Spol : </b>{{ $studentonly->spol }}</p>
        <p class="card-text"><b>Datum rođenja :</b> {{ $studentonly->datum_rodenja }}</p>
        <p class="card-text"><b>Prebivalište: </b>{{ $studentonly->adresa_prebivalista }} ({{$studentonly->mjesto->opcina?->naziv_opcine}},
        {{$studentonly->mjesto->opcina->zupanija?->naziv_zupanije}}, {{$studentonly->mjesto->opcina->zupanija->drzava->naziv_drzave}})</p>
        <p class="card-text"><b>Boravište :</b> {{ $studentonly->adresa_boravista }}  ({{$studentonly->boravistemjesto->opcina?->naziv_opcine}},
        {{$studentonly->boravistemjesto->opcina->zupanija?->naziv_zupanije}}, {{$studentonly->boravistemjesto->opcina->zupanija->drzava->naziv_drzave}})</p>
        <p class="card-text"><b>Mjesto rođenja: </b>{{ $studentonly->rodenje->naziv_mjesta }}</p>
        <p class="card-text"><b>Državljansto :</b> {{ $studentonly->drzavljanstvo->naziv_drzavljanstva }}</p>
        <p class="card-text"><b>Narodnost :</b> {{ $studentonly->narodnost->naziv_narodnosti }}</p>
        <p class="card-text"><b>Mail :</b> {{ $studentonly->email }}</p>
        <p class="card-text"><b>Mobitel :</b> {{ $studentonly->mobitel }}</p>
        <p class="card-text"><b>Telefon :</b> {{ $studentonly->telefon }}</p>
        <p class="card-text"><b>Prosjek:</b> {{ $studentonly->prosjek }}</p></br>
        <a href="{{ url('/student/' . $studentonly->JMBAG . '/edit') }}" title="Uredi podatke"><button class="btn btn-primary btn-sm">
            <i class="fa fa-pencil-square-o" aria-hidden="true"></i> Uredi podatke o studentu</button></a>
        </br>
        <a  href="{{ url('/uzdrzavatelj/'  ) }}" class="btn btn-success btn-sm" style="float:right;" title="Sljedeća stranica">
        <i  class="fa fa-plus" aria-hidden="true"></i>Sljedeća stranica </a>

  </div>
        </hr>
  </div>
@stop
```

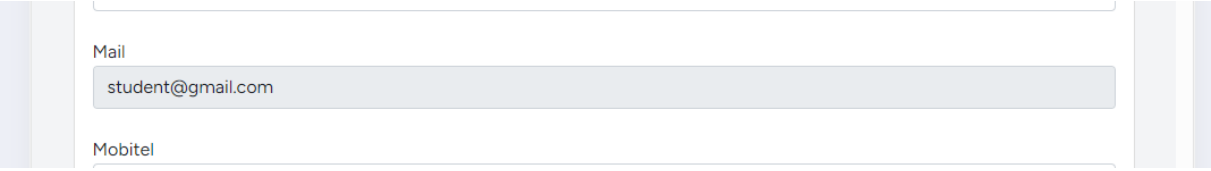
Slika 76. Prikaz index.blade.php datoteke u mapi studentonly



Slika 77. Prikaz podataka o studentu kada je korisnik prijavljen kao student



Slika 78. Prikaz ograničenja na stupac JMBAG

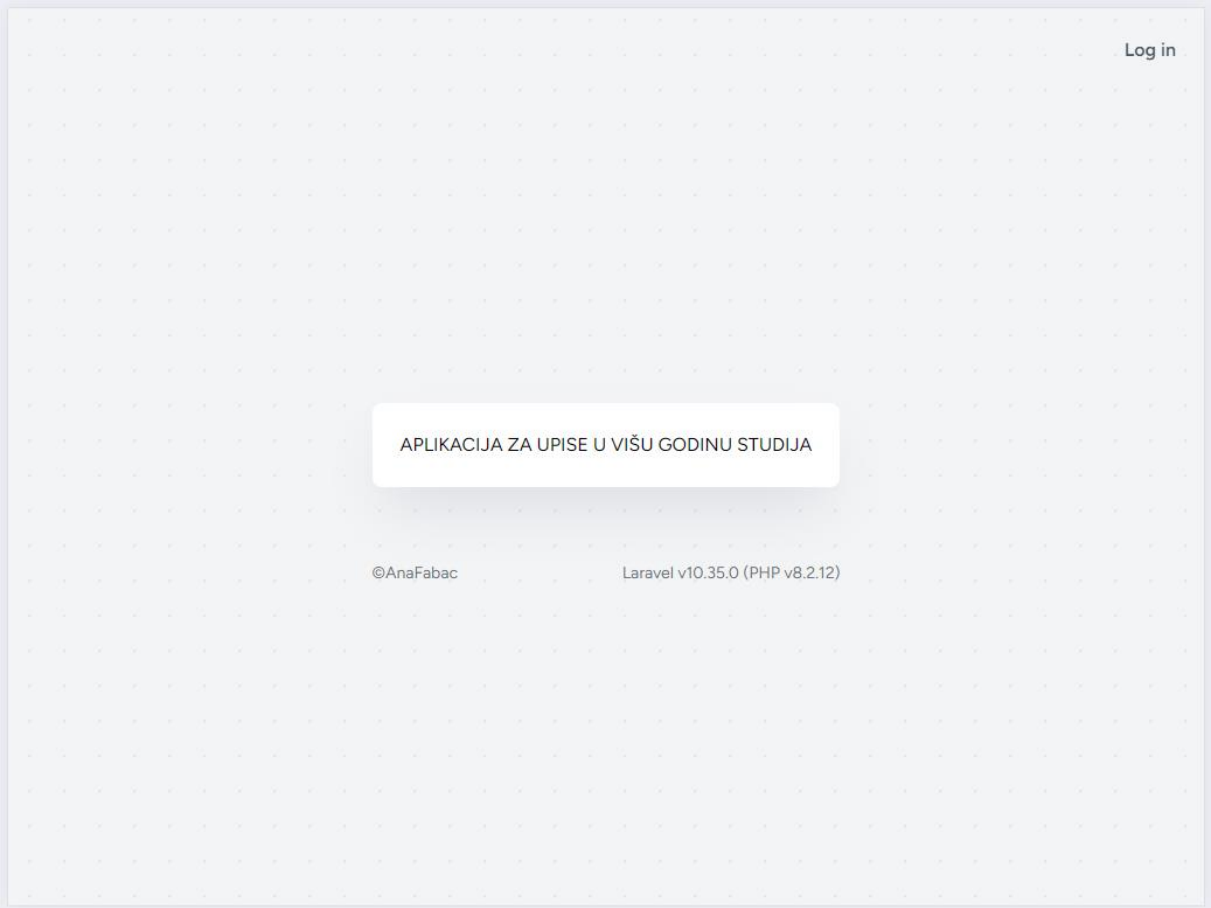


Slika 79. Prikaz ograničenja na stupac email

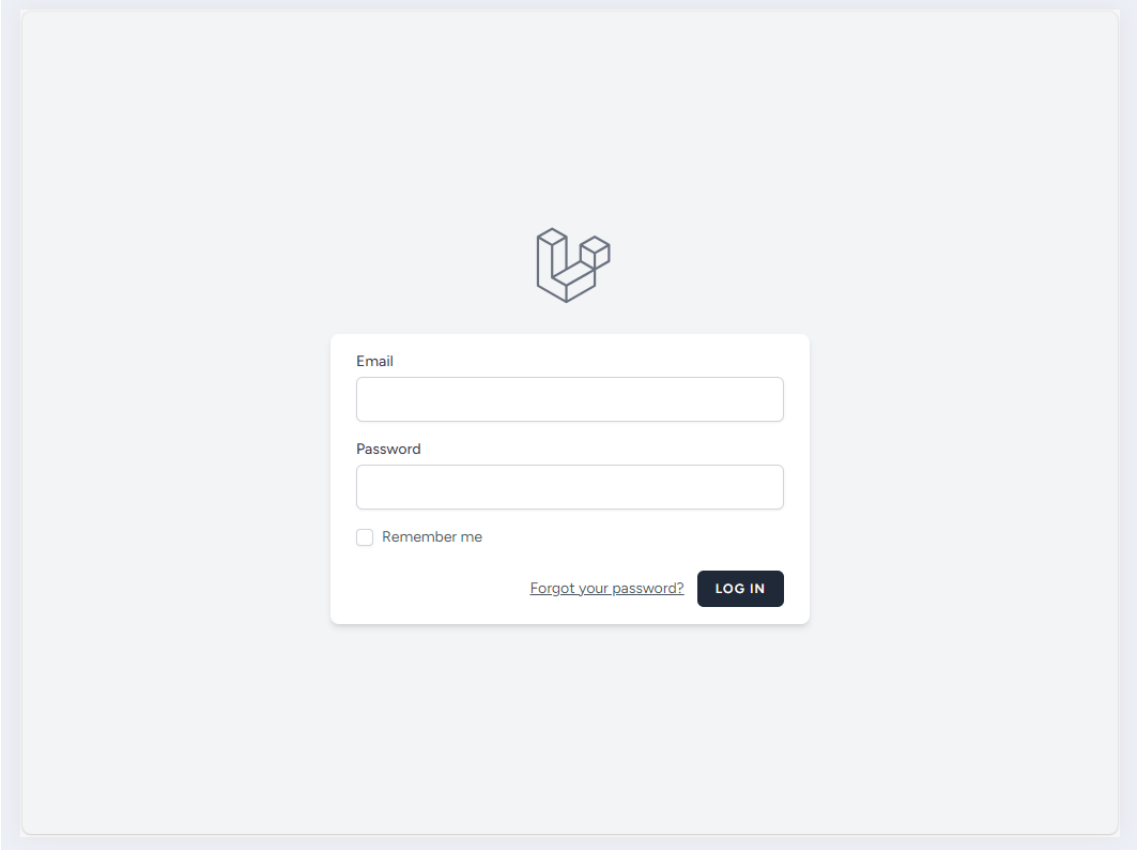
Za tablicu *upisi* potrebno je ograničiti prikaz stupca *JMBAG* kao i kod tablice *student* pošto se upisi odnose na studenta koji je prijavljen u sustav, dok za tablicu *kolegij_student* potrebno je samo prikazati kolegije koji pripadaju studentu bez da student može izvršavati akcije nad njima. Pisanje koda za tablice *upisi* i *kolegij_student* je po istom principu koji je opisan kod tablica *student* i *uzdrzavatelj*.

6. PRIKAZ GOTOVE APLIKACIJE

U ovom poglavlju proći će se kroz tijek aplikacije za korisnike čija je uloga student i admin. Kako bi se korisnik mogao prijaviti u aplikaciju potrebno je da u bazi podataka postoji stvoren račun za korisnika. Dodavanje novih korisnika planirano je kroz direktni import podataka u bazu podataka. Nakon dolaska na početnu stranicu aplikacije korisnik može kliknuti na tekst „Log in“ koji se nalazi u gornjem desnom kutu stranice. Nakon čega se prikazuje ekran za *Log in* u aplikaciju kao što je prikazano na Slici 81.

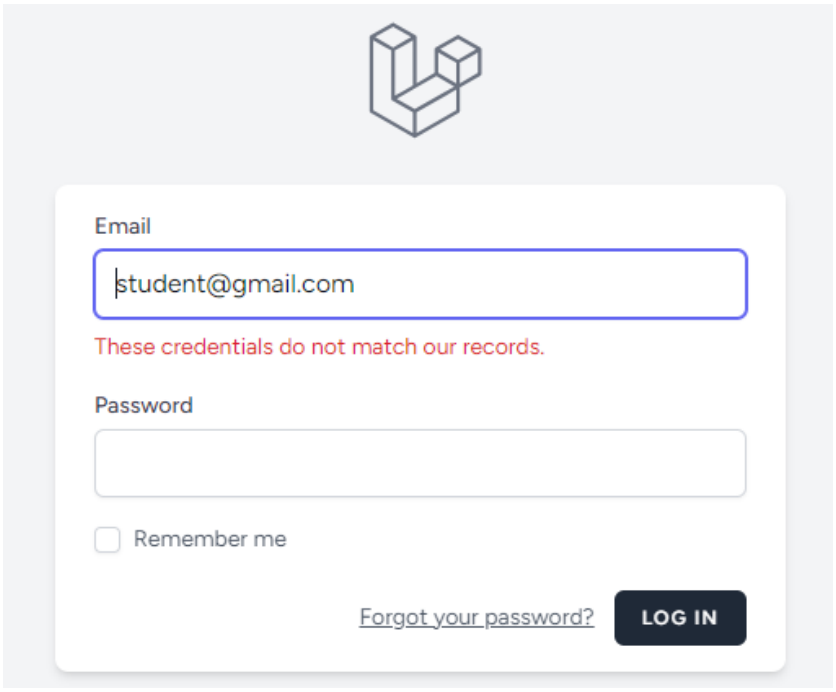


Slika 80. Prikaz početnog ekrana



Slika 81. Prikaz ekrana za Log in

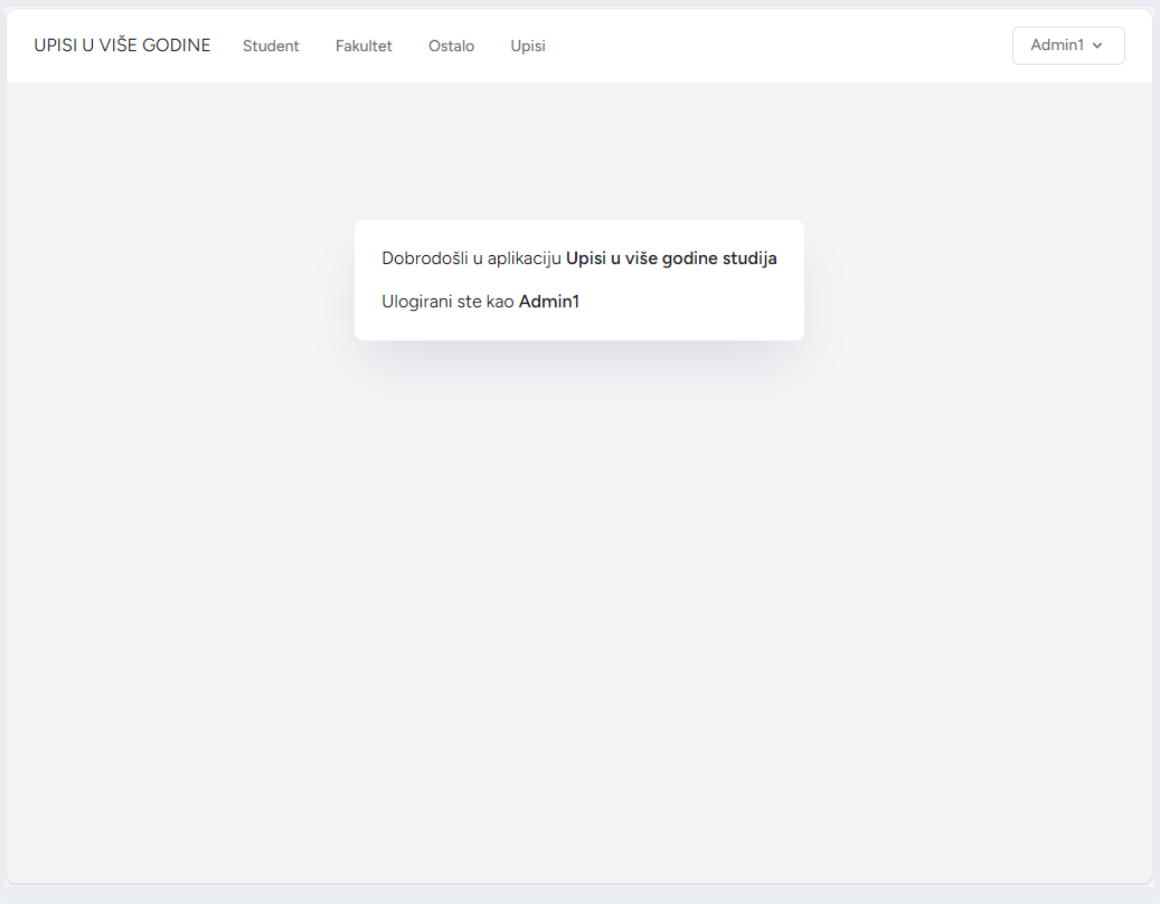
Nakon pojave ekrana korisnik upisuje email i lozinku te ukoliko se podaci ne podudaraju s podacima u bazi podataka korisniku će se ispisati odgovarajuća poruka (Slika 82). Nakon uspješnog logina korisnika na temelju njegove uloge preusmjeri na odgovarajuću stranicu.



Slika 82. Prikaz pokušaja logina

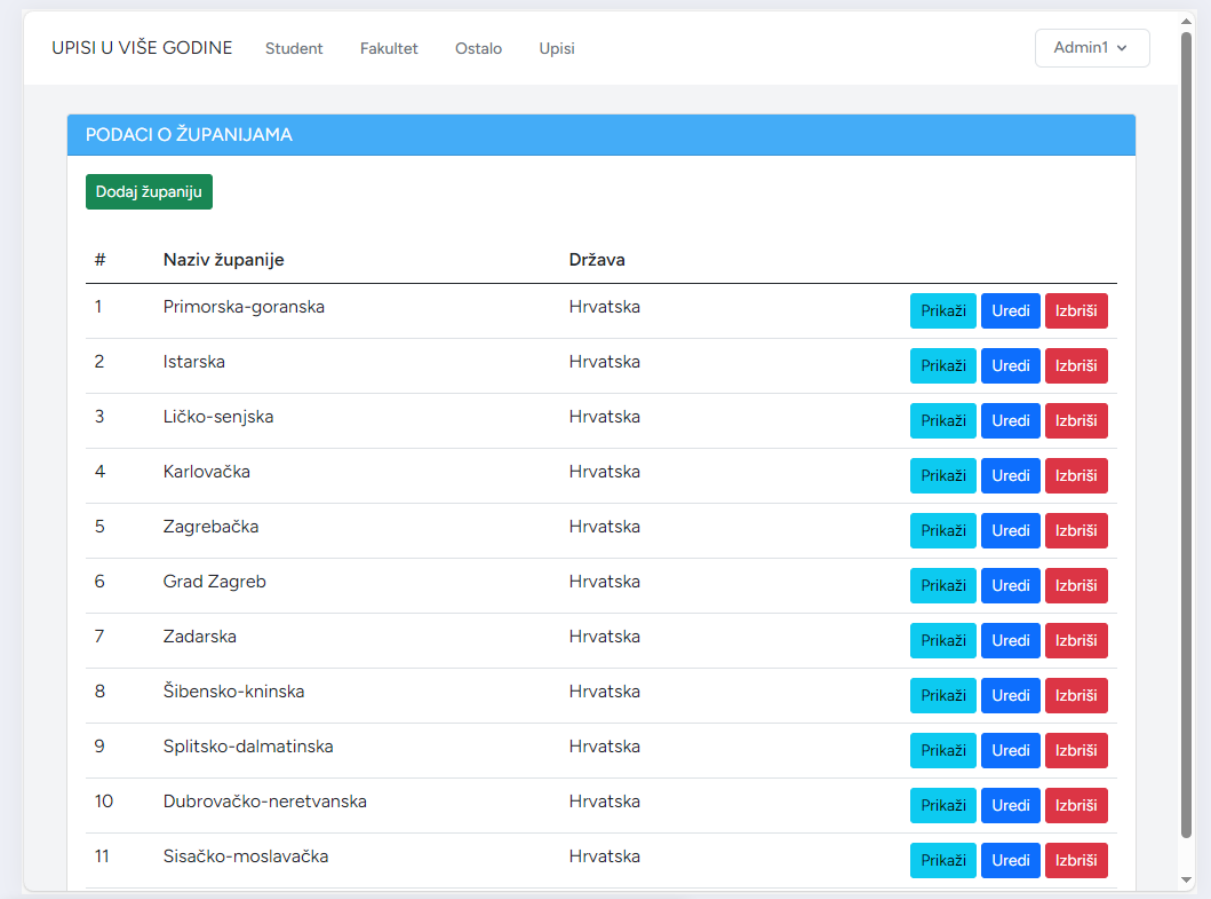
6.1. TIJEK APLIKACIJE KADA JE ULOGA KORISNIKA ADMIN

Ukoliko se je korisnik prijavljen kao admin u aplikaciju prikazuje mu se ekran na kojem se ispisuje da je prijavljen kao admin. Nakon čega korisnik može početi s unosom, pregledom, ažuriranja ili brisanje podataka iz odabranih tablica. Admin ima pristup svim tablica te na svim tablicama može izvršiti sve prethodno navede akcije.

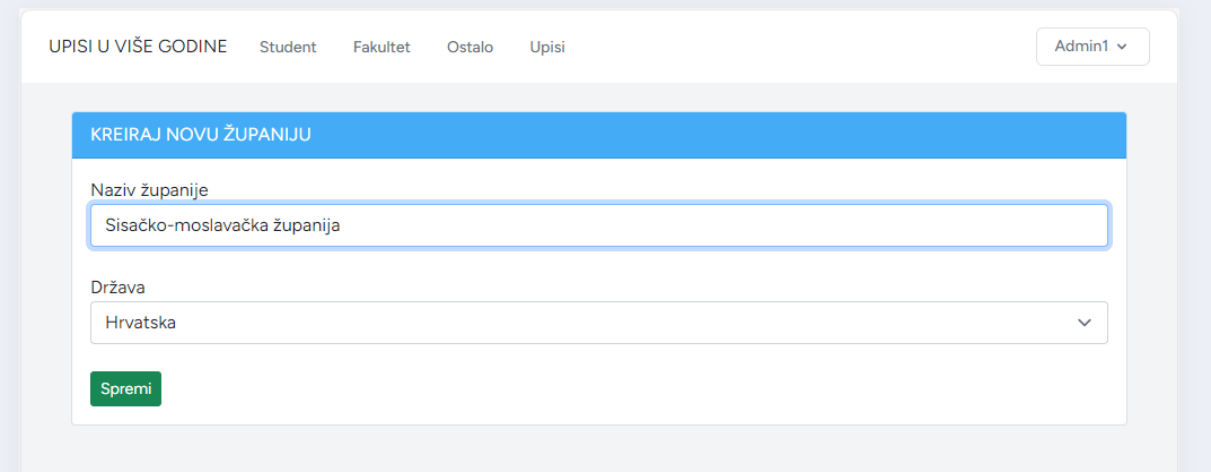


Slika 83. Prikaz ekrana nakon logina kao admin

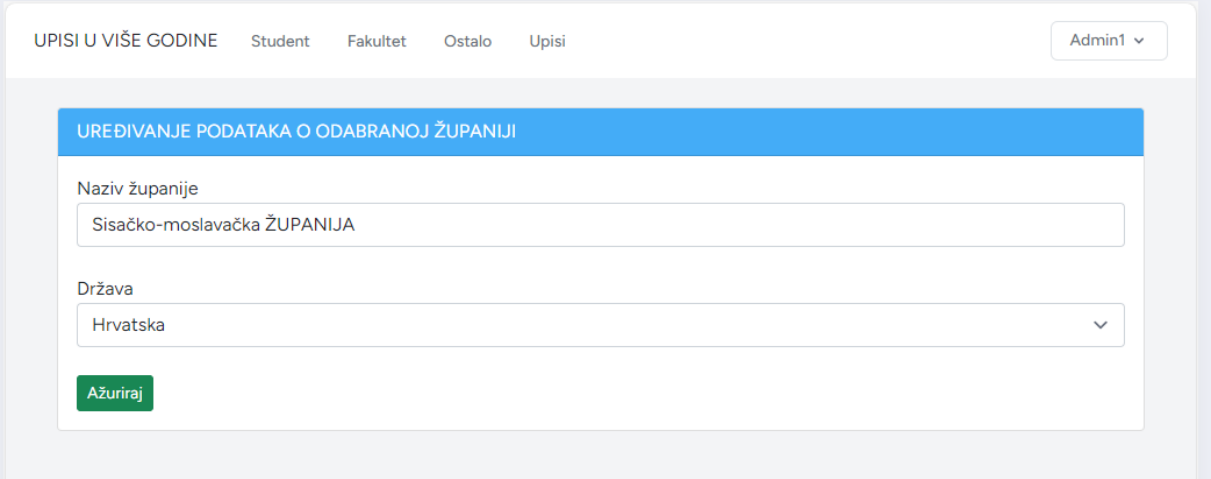
Klikom na Ostalo u navigaciji otvara se padajući izbornik iz kojeg je moguće izabrati *Država*, *Županija*, *Općina*, *Mjesto*, *Državljanstvo*, *Narodnost*, *Obrazovanje*, *Stručna sprema*, *Zanimanje*, *Položaj u zanimanju* te odabirom Županija otvara se prikaz svih unesenih županija. Klikom na „Dodaj županiju“ otvara forma za kreiranje nove županije u koju korisnik unosi naziv županije, nakon spremanja županija se prikazuje na popisu županija te moguće ju je urediti klikom na gumb „Uredi“. Nakon uređivanja podataka potrebno je kliknuti na gumb „Ažuriraj“ kako bi se podaci ažurirali te nakon čega se korisnika vrati na listu županija. Također, moguće je prikazi sve podatke za svaku pojedinačnu županiju na način da se klikne na gumb „Prikaži“. Prikaz u web pregledniku prikazan je na Slici 84., Slici 85., Slici 86., Slici 87. i Slici 88.



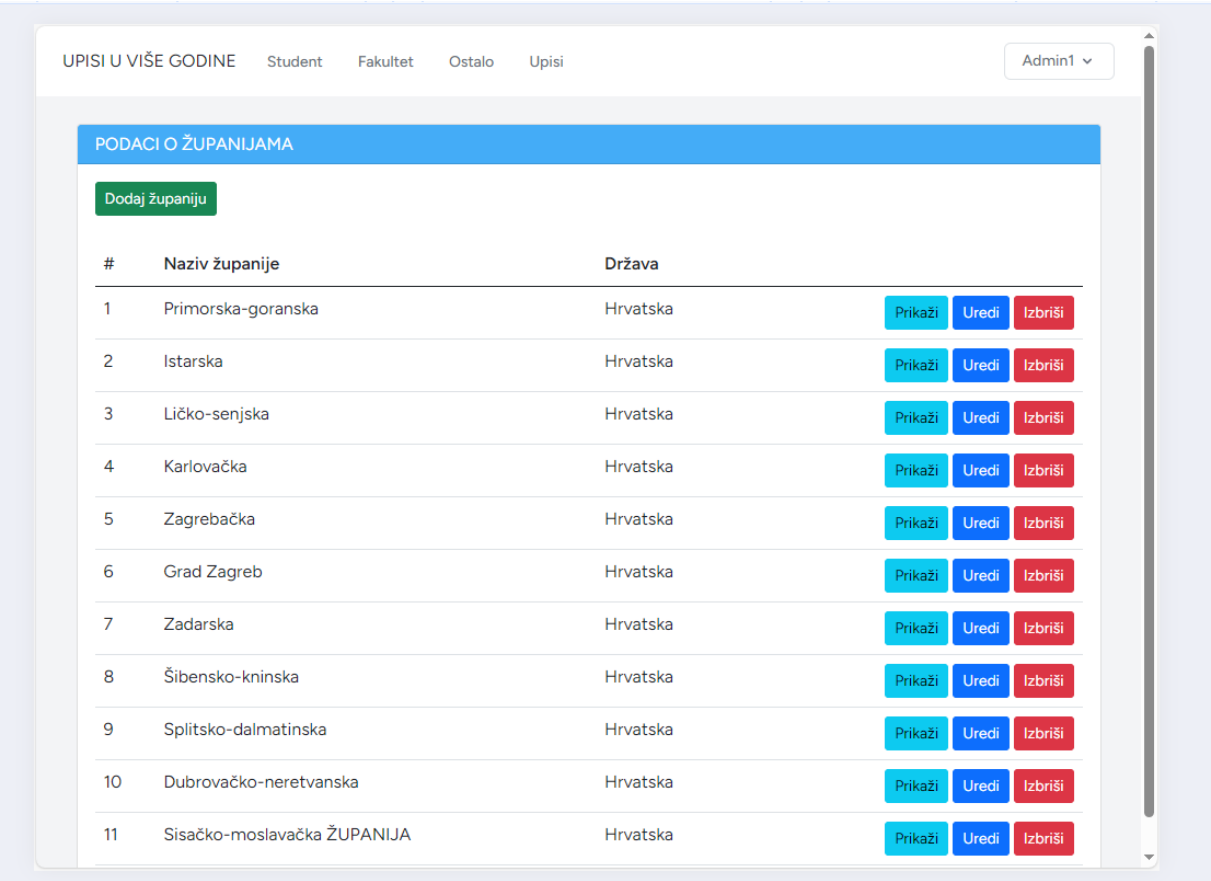
Slika 84. Prikaz liste županija



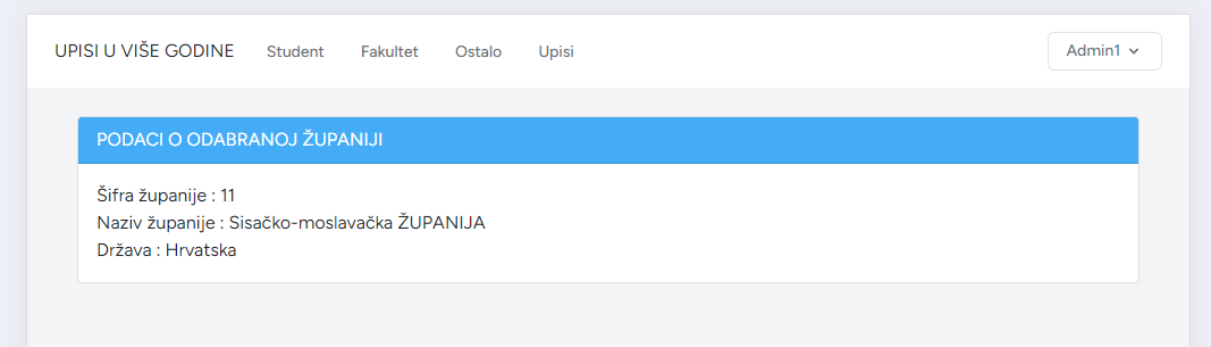
Slika 85. Prikaz forme za unos županije



Slika 86. Prikaz forme za uređivanje županije

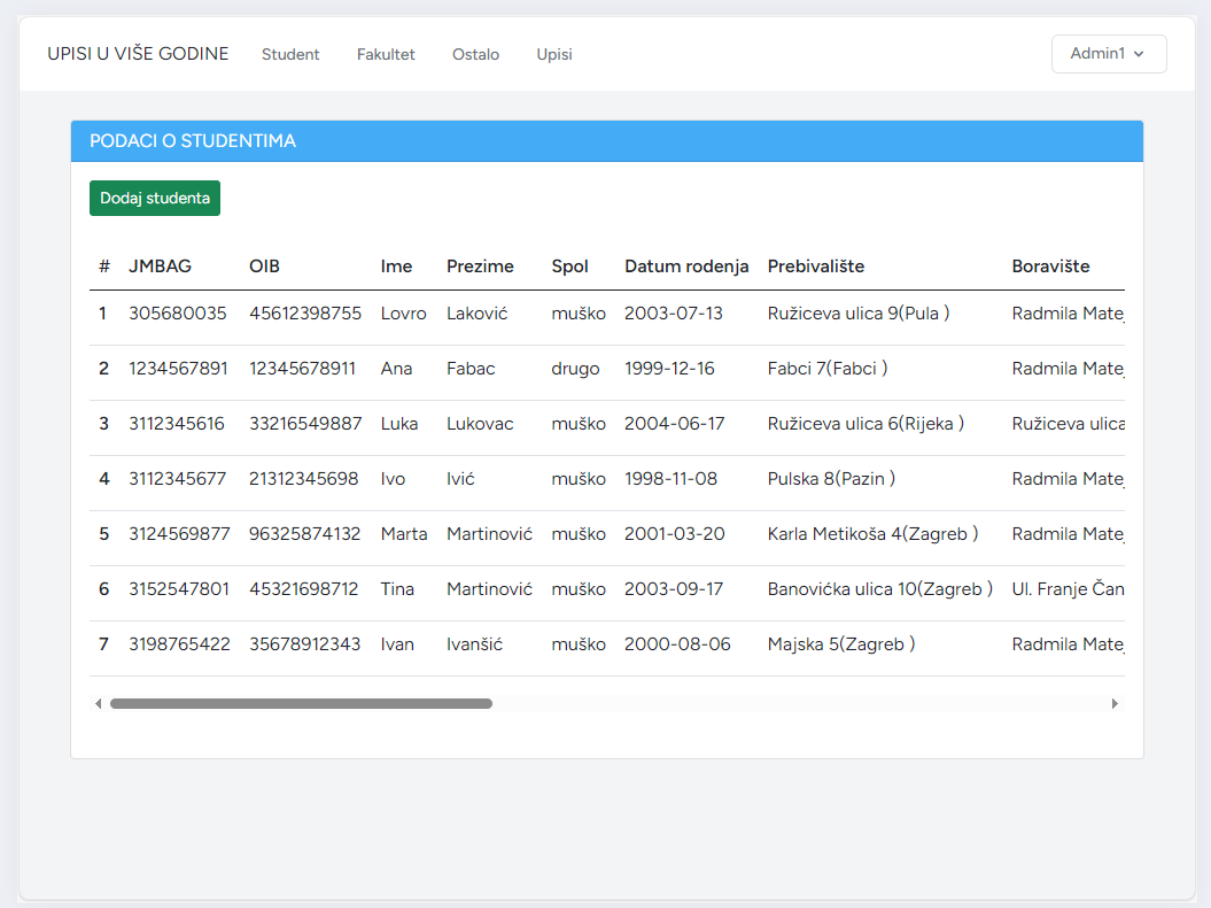


Slika 87. Prikaz liste županije nakon ažuriranja županije



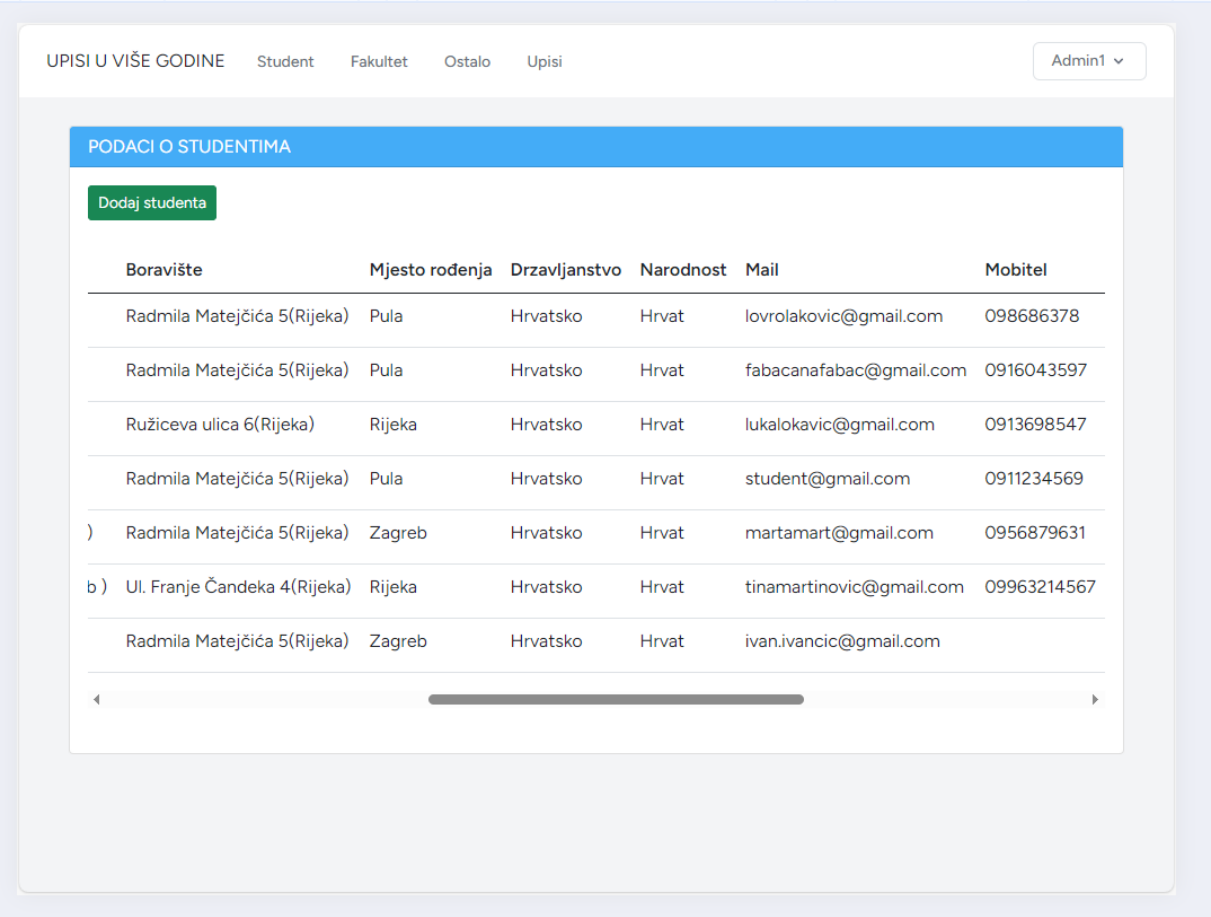
Slika 88. Prikaz podataka o odabranoj županiji

U navigaciji korisnik može izabrati tablicu *student* na način da klikne na student te iz padajućeg izbornika izabere *Student* nakon čega se korisniku prikazuje lista svih studenta. Kao i kod županije korisnik može dodani novog studenta, urediti, pregledati podatke ili izbrisati studenta. Na Slici 89., Slici 90. i Slici 91. prikazana je lista svih studenta s svim podacima.

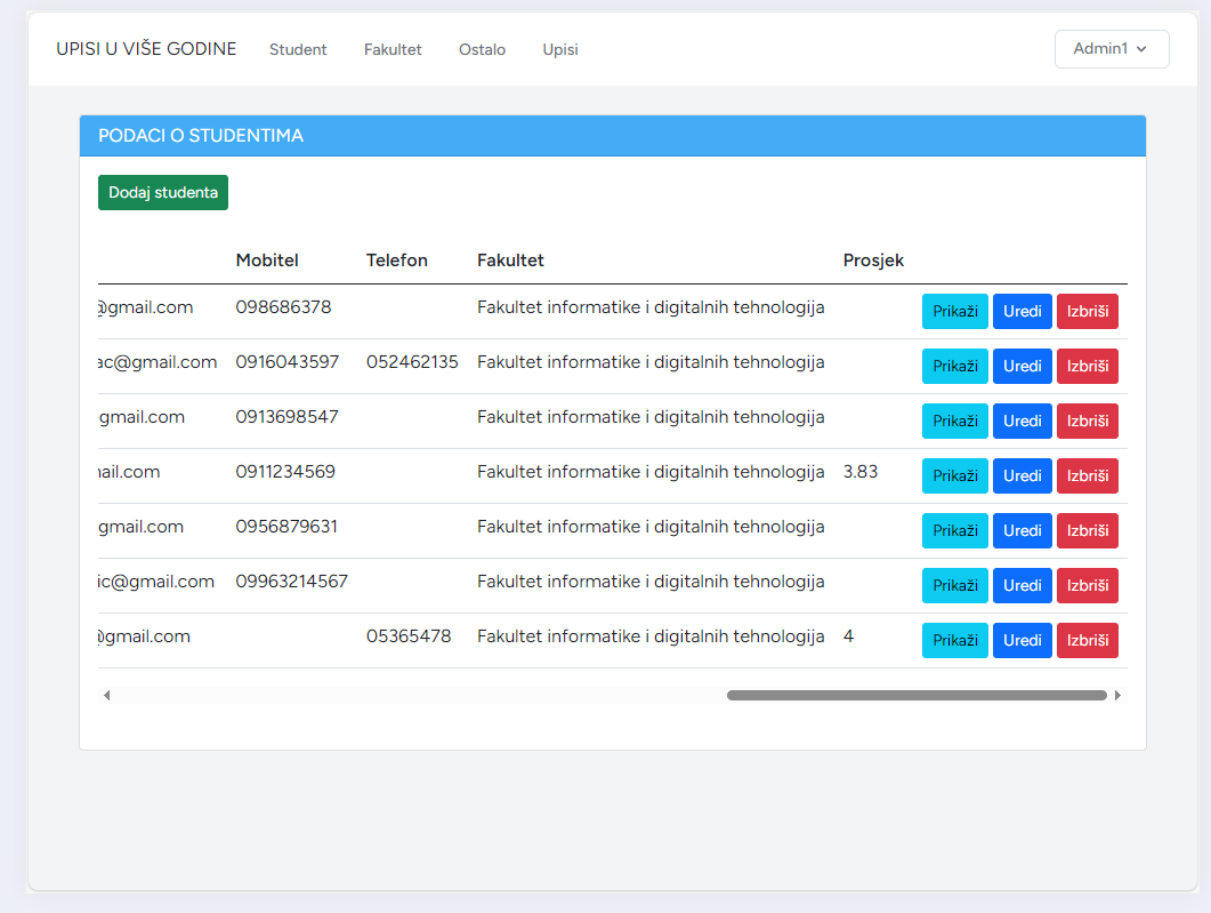


#	JMBAG	OIB	Ime	Prezime	Spol	Datum rodenja	Prebivalište	Boravište
1	305680035	45612398755	Lovro	Laković	muško	2003-07-13	Ružiceva ulica 9(Pula)	Radmila Mate
2	1234567891	12345678911	Ana	Fabac	drugo	1999-12-16	Fabci 7(Fabci)	Radmila Mate
3	3112345616	33216549887	Luka	Lukovac	muško	2004-06-17	Ružiceva ulica 6(Rijeka)	Ružiceva ulica
4	3112345677	21312345698	Ivo	Ivić	muško	1998-11-08	Pulska 8(Pazin)	Radmila Mate
5	3124569877	96325874132	Marta	Martinović	muško	2001-03-20	Karla Metikoša 4(Zagreb)	Radmila Mate
6	3152547801	45321698712	Tina	Martinović	muško	2003-09-17	Banovićka ulica 10(Zagreb)	Ul. Franje Čan
7	3198765422	35678912343	Ivan	Ivanšić	muško	2000-08-06	Majska 5(Zagreb)	Radmila Mate

Slika 89. Prikaz liste svih studenata 1. dio

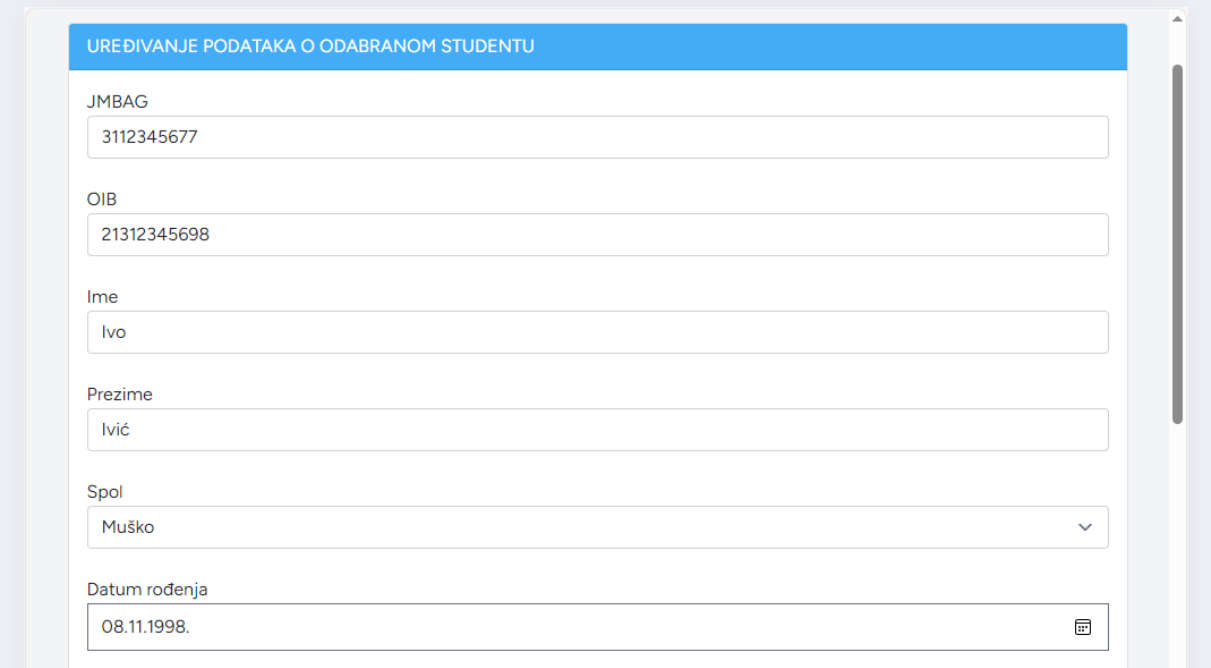


Slika 90. Prikaz liste svih studenata 2. dio



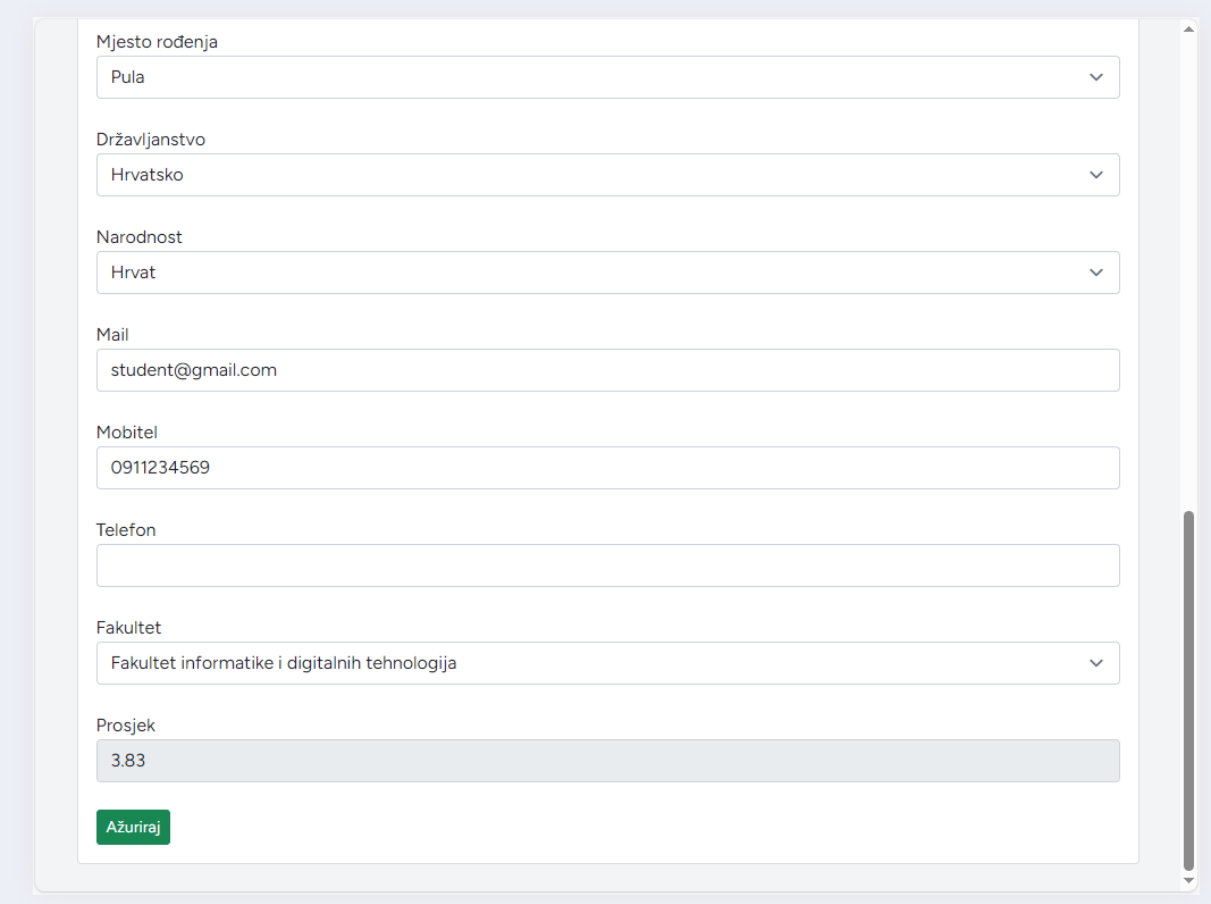
Slika 91. Prikaz liste svih studenata 3. dio

Kod forme za uređivanje podataka o studentu korisnik može uređivati sva polja osim prosjeka koji se sam ažurira dodavanjem kolegija studentu te sva polja za koje već postoje podaci imaju ponuđen padajući izbornik s podacima koji su spremljeni u povezanim tablicama ili unaprijed definirani. Padajući izbornici u formi za uređivanje podataka o studentu imaju polja spol, mjesto prebivališta, mjesto boravišta, mjesto rođenja, državljanstvo, narodnost i fakultet. Prikaz forme u web pregledniku prikazano je na Slici 92., Slici 93. i Slici 94.



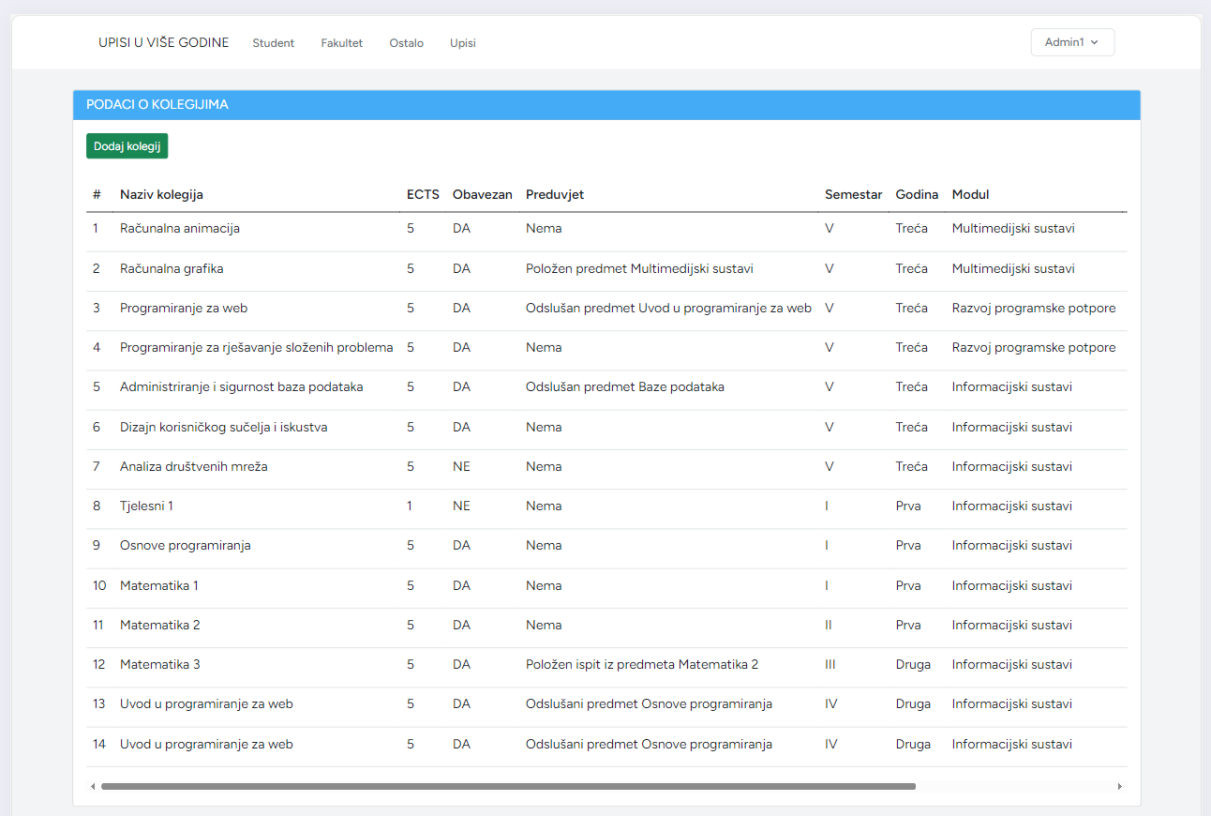
Slika 92. Prikaz forme uređivanja studenta 1.dio

Slika 93. Prikaz forme uređivanja studenta 2.dio



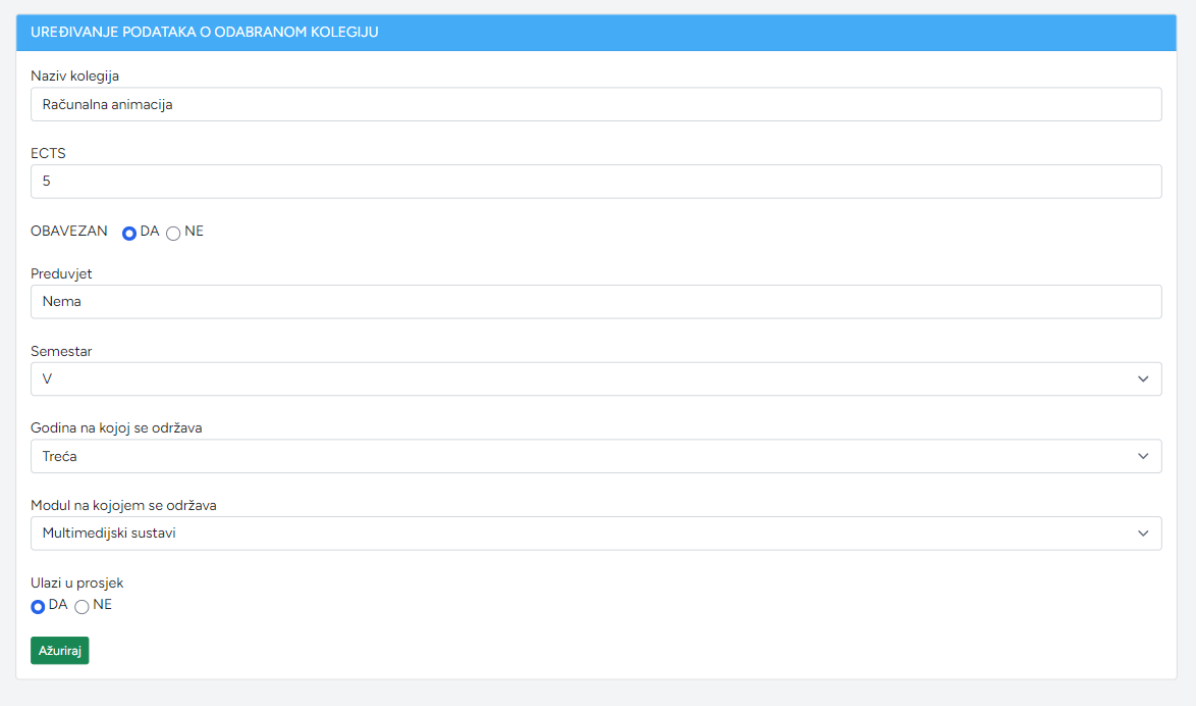
Slika 94. Prikaz forme uređivanja studenta 3.dio

U navigaciji korisnik može izabrati tablicu *kolegij* na način da klikne na *Fakultet* te iz padajućeg izbornika te je potrebno izabrati *Kolegij* kako bi se korisniku prikazala lista svih kolegija. Kao i kod tablice *student* korisnik može dodati novi kolegij, urediti i pregledati podatke o kolegiju ili izbrisati kolegij. Kod kreiranja ili dodavanja kolegija prikazuje se radio dugme za polja obavezan i ulazi u prosjek, dok padajući izbornik je implementiran za polja semestar, godina na kojoj se odražava i modul. Na Slici 95. je prikazana lista svih kolegija dok na Slici 96. je prikazan forma za uređivanje podataka o kolegiju.



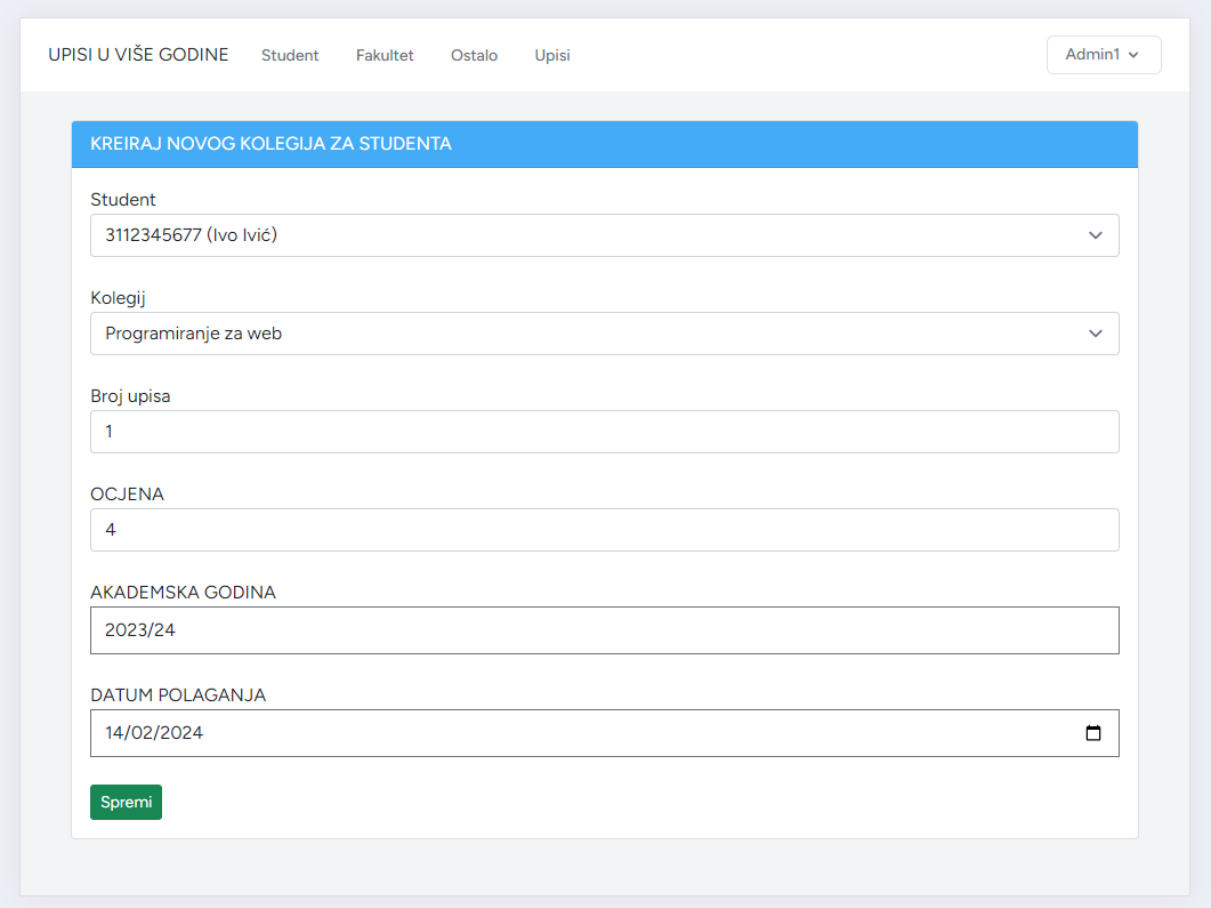
#	Naziv kolegija	ECTS	Obavezan	Preduvjet	Semestar	Godina	Modul
1	Računalna animacija	5	DA	Nema	V	Treća	Multimedijski sustavi
2	Računalna grafika	5	DA	Položen predmet Multimedijski sustavi	V	Treća	Multimedijski sustavi
3	Programiranje za web	5	DA	Odslušan predmet Uvod u programiranje za web	V	Treća	Razvoj programske potpore
4	Programiranje za rješavanje složenih problema	5	DA	Nema	V	Treća	Razvoj programske potpore
5	Administriranje i sigurnost baza podataka	5	DA	Odslušan predmet Baze podataka	V	Treća	Informacijski sustavi
6	Dizajn korisničkog sučelja i iskustva	5	DA	Nema	V	Treća	Informacijski sustavi
7	Analiza društvenih mreža	5	NE	Nema	V	Treća	Informacijski sustavi
8	Tjelesni 1	1	NE	Nema	I	Prva	Informacijski sustavi
9	Osnove programiranja	5	DA	Nema	I	Prva	Informacijski sustavi
10	Matematika 1	5	DA	Nema	I	Prva	Informacijski sustavi
11	Matematika 2	5	DA	Nema	II	Prva	Informacijski sustavi
12	Matematika 3	5	DA	Položen ispit iz predmeta Matematika 2	III	Druga	Informacijski sustavi
13	Uvod u programiranje za web	5	DA	Odslušani predmet Osnove programiranja	IV	Druga	Informacijski sustavi
14	Uvod u programiranje za web	5	DA	Odslušani predmet Osnove programiranja	IV	Druga	Informacijski sustavi

Slika 95. Prikaz liste svih kolegija

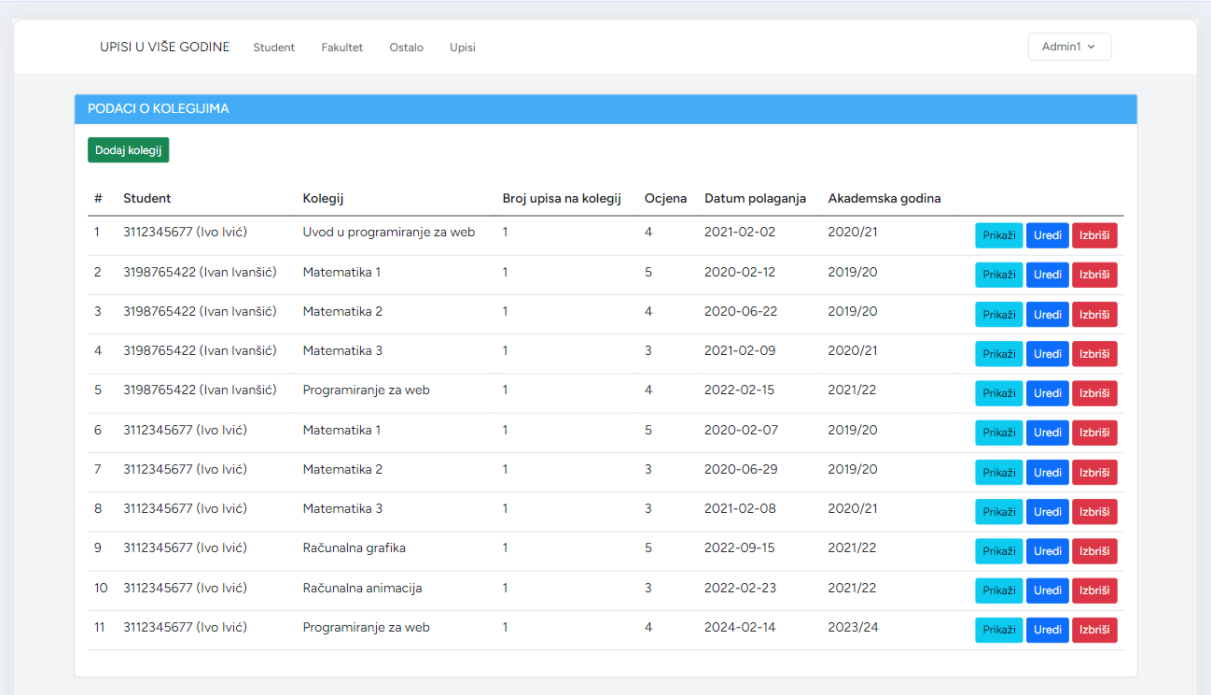


Slika 96. Prikaza forme za uređivanje podataka o kolegiju

Također, kako bi korisnik mogao dodati kolegij studentu koji nije u bazi podataka prvo mora kreirati novog studenta pomoću forme za kreiranje studenta te nakon toga provjeriti ukoliko kolegij koji želi dodati već postoji u bazi. Ako postoji u tablici *kolegij* onda može nastaviti na tablicu *kolegij_student,* no ukoliko ne postoji prvo je potrebno kreirati novi kolegij u tablici *kolegij*. Nakon čega je moguće u tablici *kolegij_student* preko forme za unos novog kolegija za studenta dodati relaciju te izabrati iz padajućeg izbornika novi kreirani kolegij i novo kreirani student i upisati ostale podatke. Prikaz kreiranje novog stupca u tablici *kolegij_student* je prikazan na Slici 97., dok lista svih stupaca iz tablice *kolegij_student* tablice je prikazana na Slici 98.



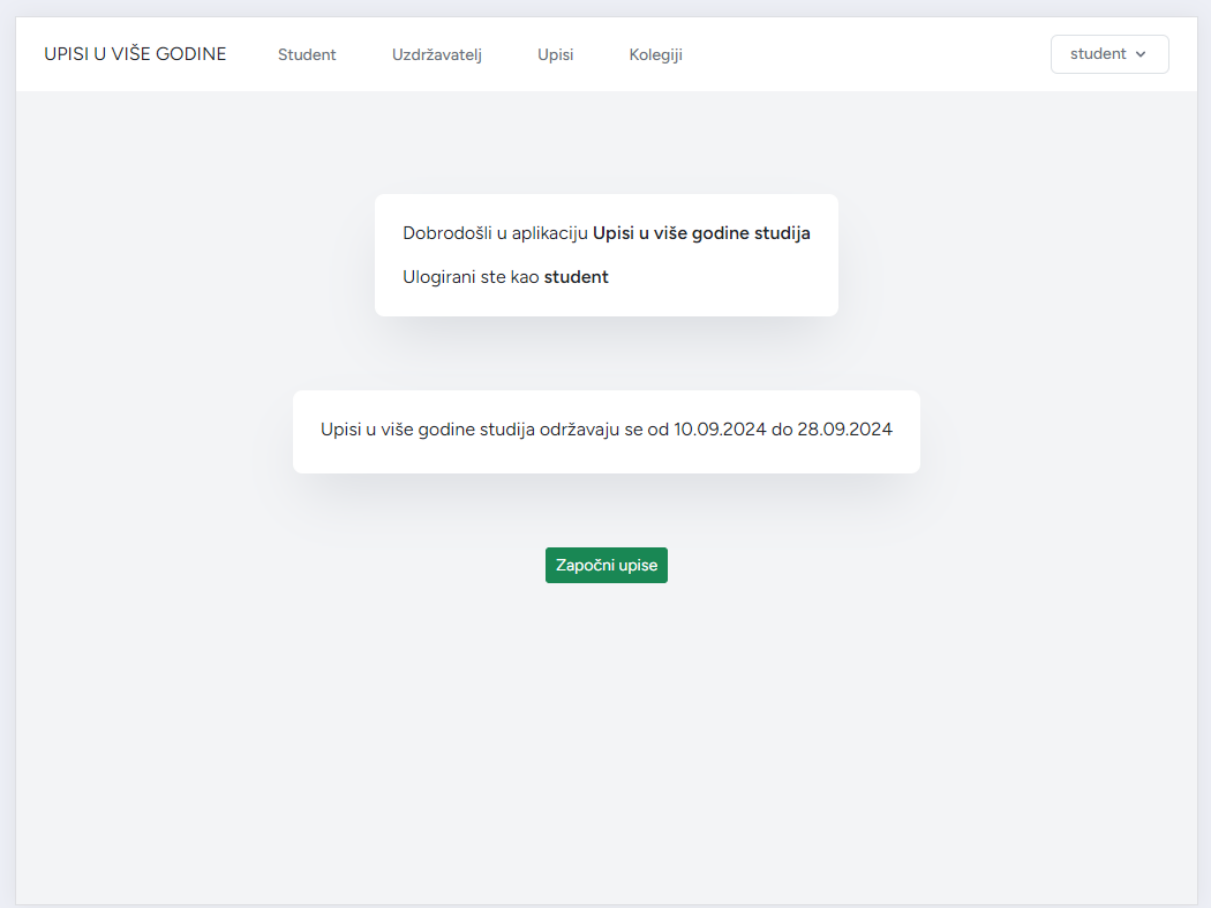
Slika 97. Prikaz forme za kreiranja novog stupca u kolegij_student *tablici*



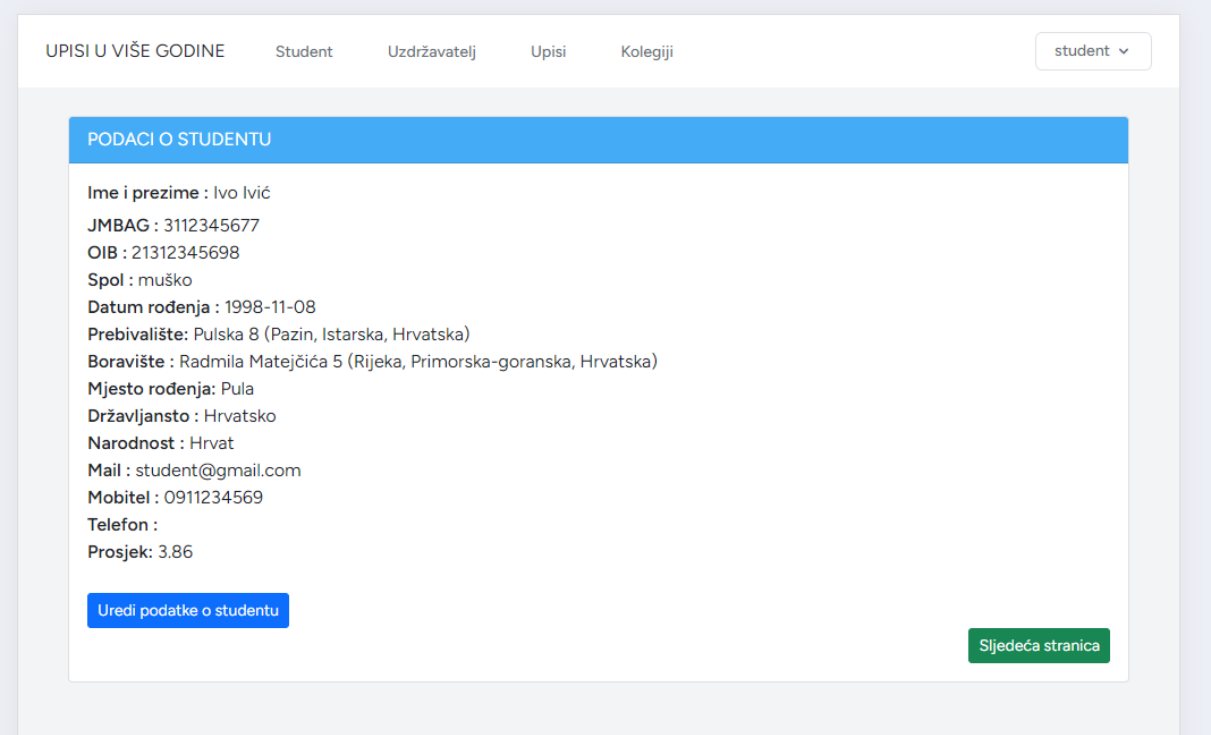
Slika 98. Prikaz lista stupaca iz tablice kolegij_student

6.2. TIJEK APLIKACIJE KADA JE ULOGA KORISNIKA STUDENT

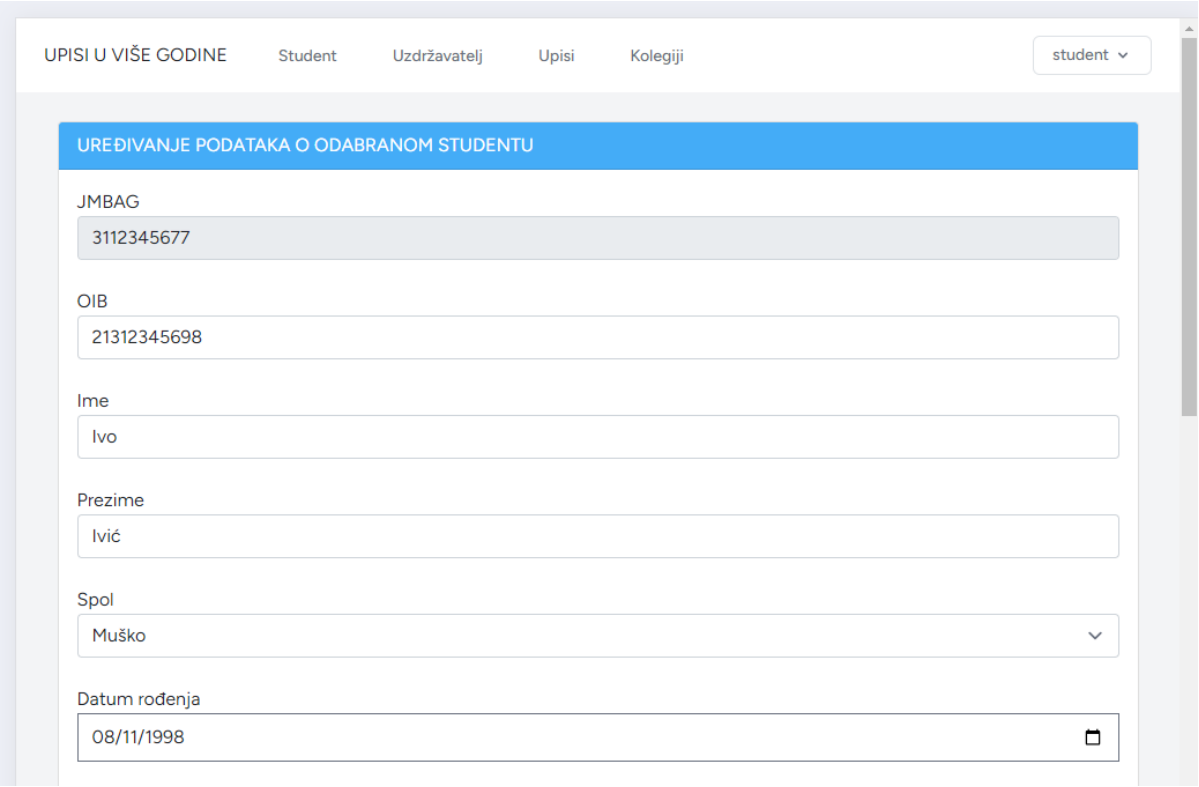
Ukoliko se je korisnik prijavljen kao student u aplikaciju prikazuje mu se ekran na kojem se ispisuje da je prijavljen kao student uz gumb *„Započni upise“*. Nakon čega korisnik može početi s procesom upisa preko gumba ili preko navigacije. Klikom na gumb studenta se vodi na stranicu s prikazom podataka o prijavljenom studentu i gumbom „*Uredi podatke*“ s kojim na klik otvara formu za uređivanja podataka o student u kojoj student ne može uređivati JMBAG, mail i prosjek kao što je prikazano na Slici 101 i Slici 103. Te ukoliko student ne želi urediti podatke može kliknuti na gumb *„Sljedeća stranica“* ili *Uzdržavatelj* u navigacije čime studenta vodi na stranicu o njegovim uzdržavateljima.



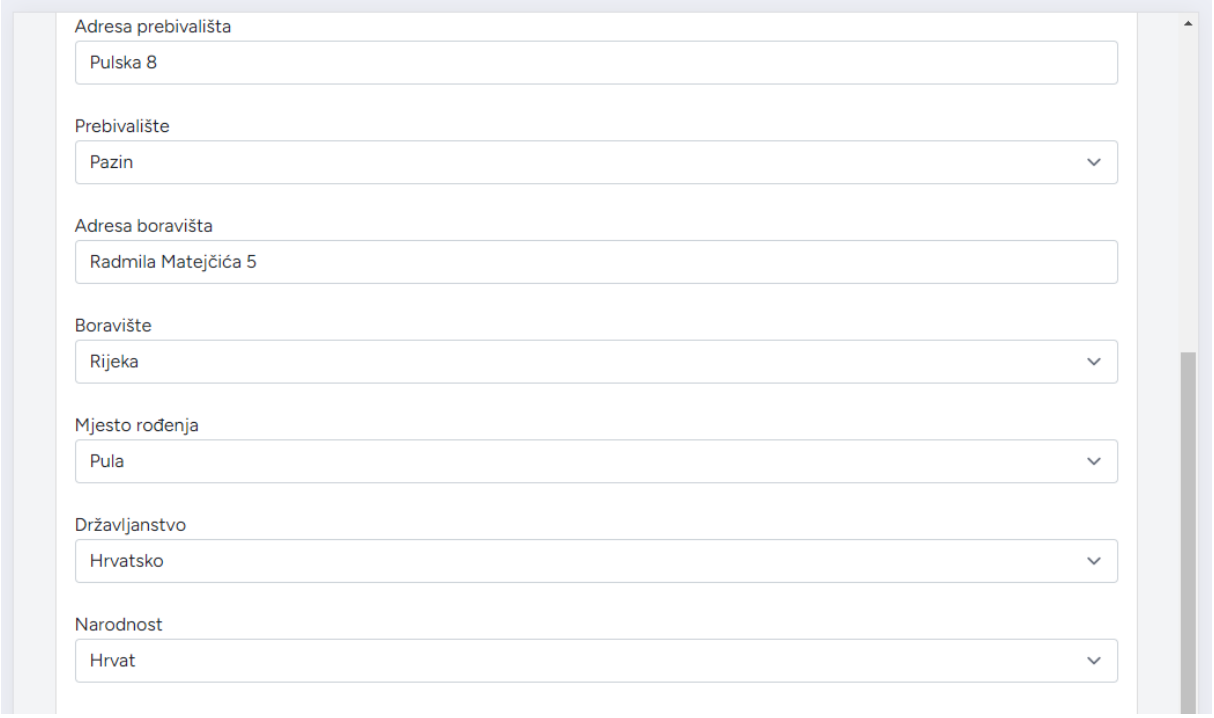
Slika 99. Prikaz ekrana nakon logina kao student



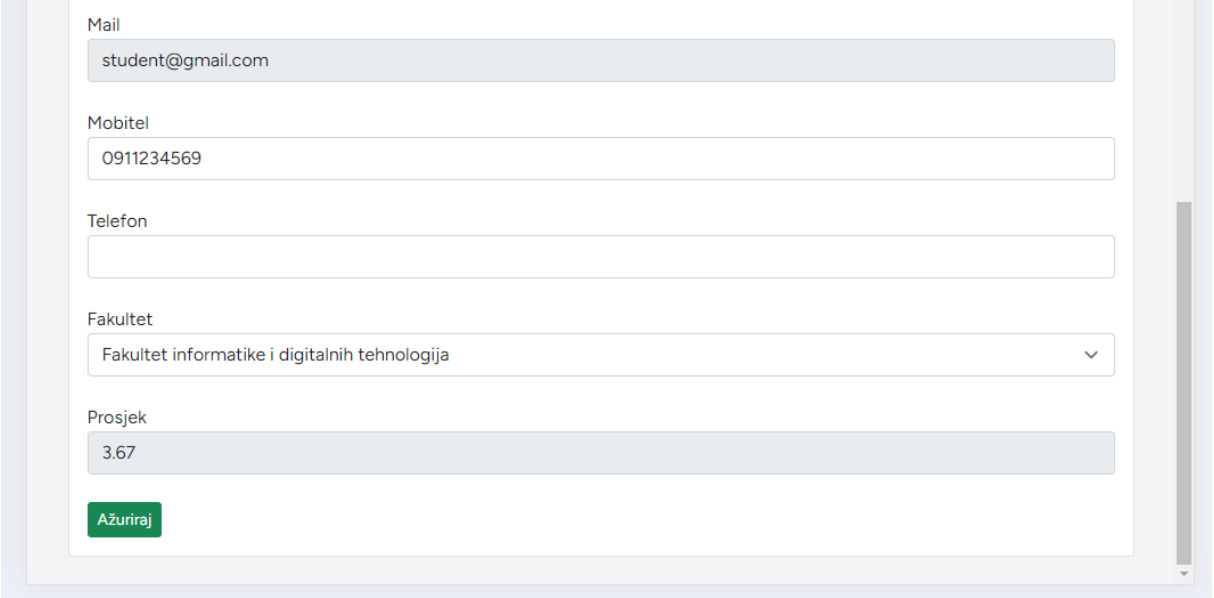
Slika 100. Prikaz podataka o prijavljenom studentu



Slika 101. Prikaz uređivanja podataka o studentu 1.dio

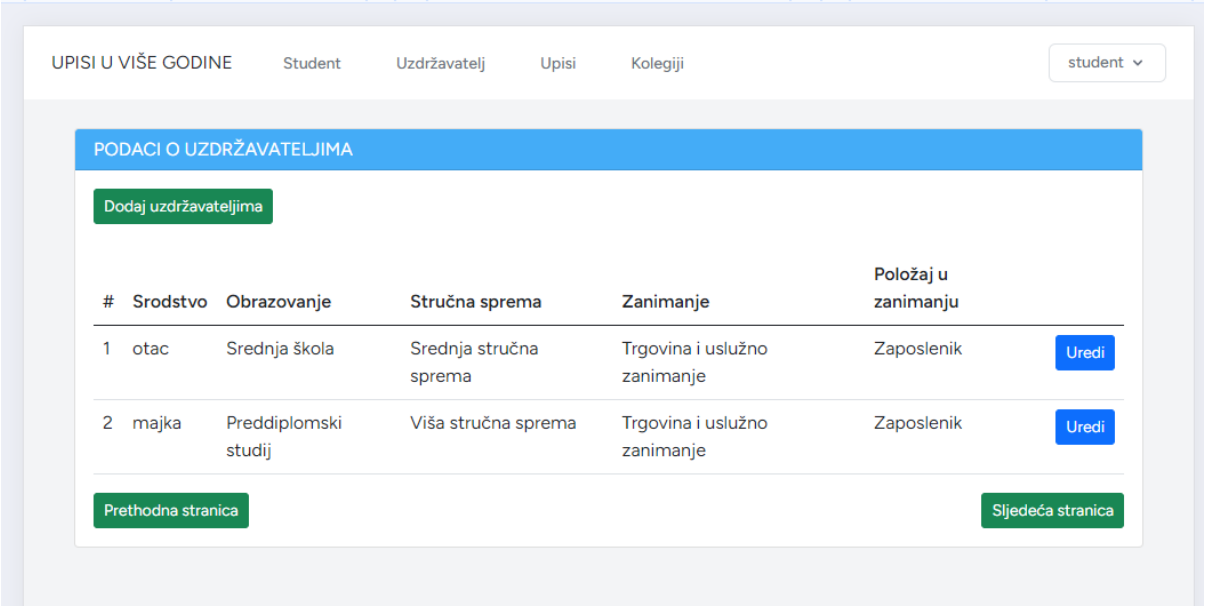


Slika 102. Prikaz uređivanja podataka o studentu 2.dio

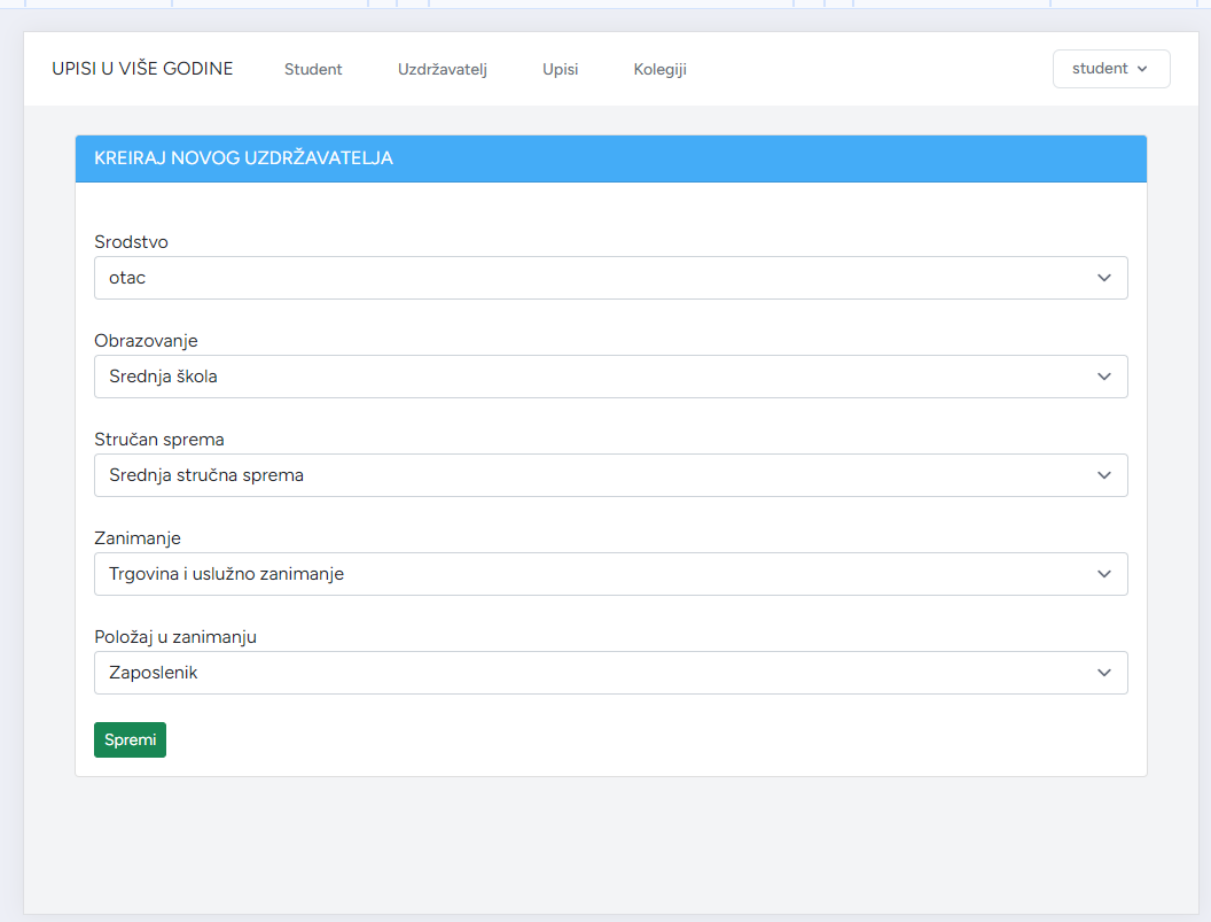


Slika 103. Prikaz uređivanja podataka o studentu 3.dio

Nakon prikaza stranice o uzdržavateljima student može ili urediti postojeće uzdržavatelje ili dodati novog pomoću gumba *„Dodaj uzdržavatelja*“ te ispunjavanje forme za kreiranje novog uzdržavatelja. Nakon uređivanja podataka student može kliknuti na gumb „Sljedeća stranica“ ili Upisi u navigacije čime studenta vodi na stranicu o njegovim upisima.

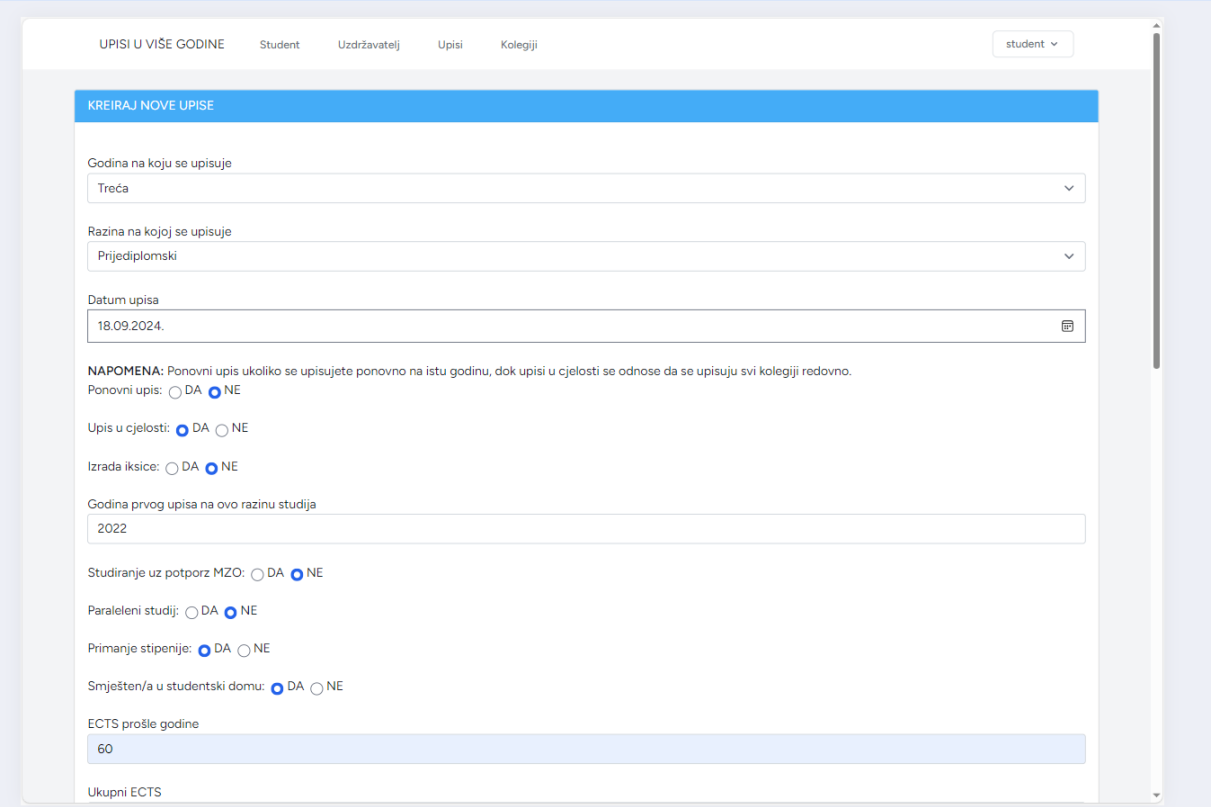


Slika 104. Prikaz liste uzdržavatelja za prijavljenog studenata

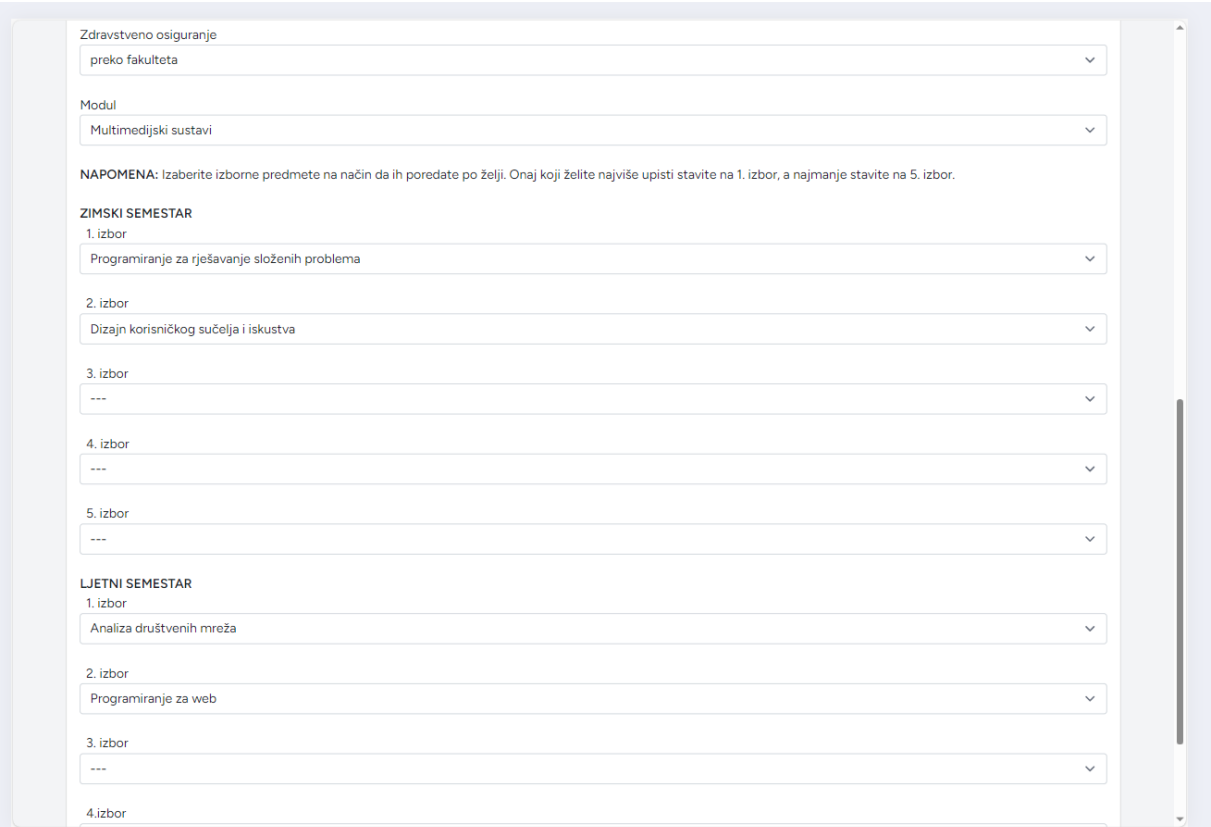


Slika 105. Prikaz forme za kreiranje novog uzdržavatelja

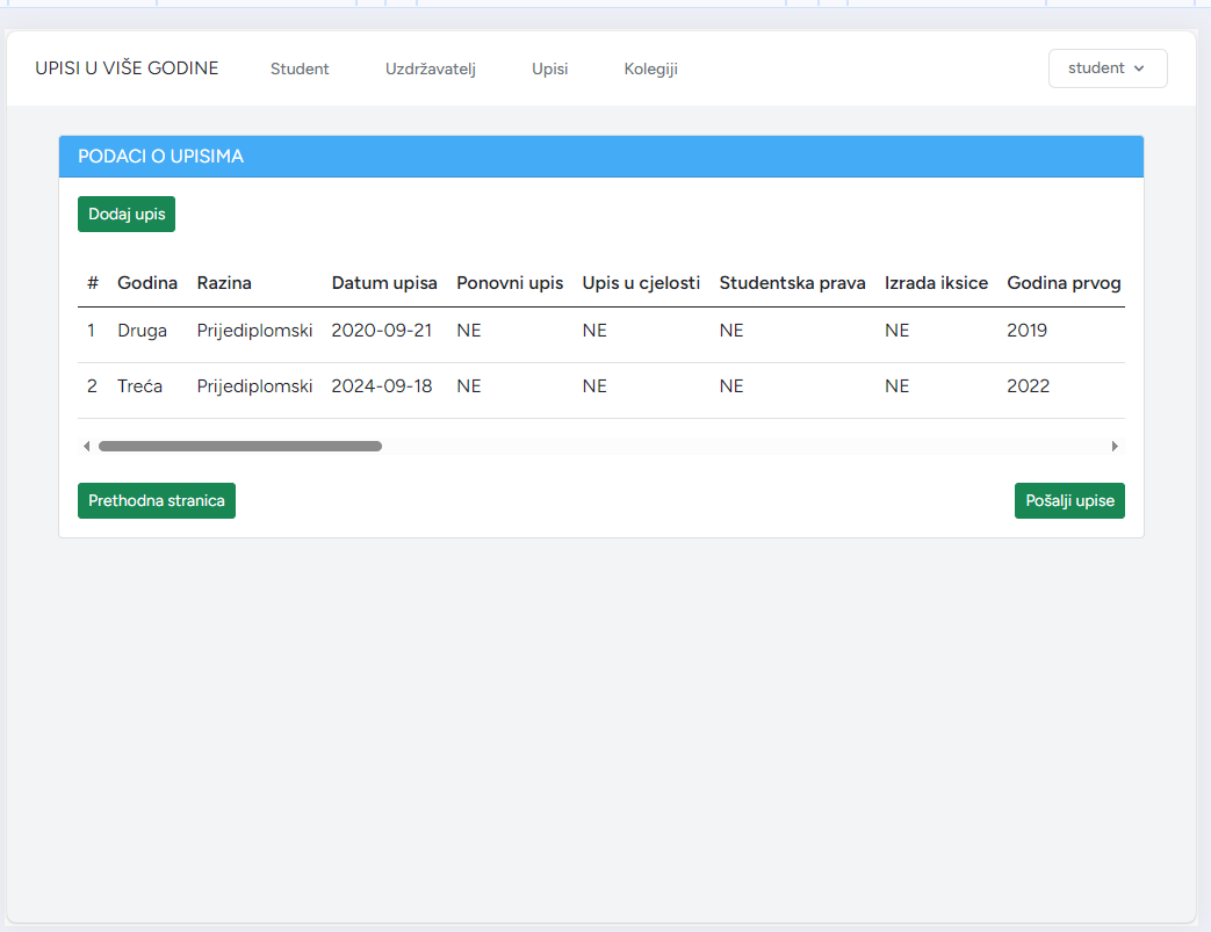
Nakon dolaska do upisa potrebno je dodati nove upise te ispuniti sve podatke unutar forme kao što je prikazano na Slici 106. i Slici 107. te nakon što se ispuni forma i izaberu izborni predmeti studenta se vrati na listu upisa. Klikom na gumb „Pošalji upise“ studenta se preusmjeri na novu stranicu na kojoj se daje informacija studentu da su upisi poslani i da će biti obaviješteni od strane studentske kada upisi budu gotovi nakon čega student saznaje ukoliko je upisao željene izborne predmete (Slika 109).



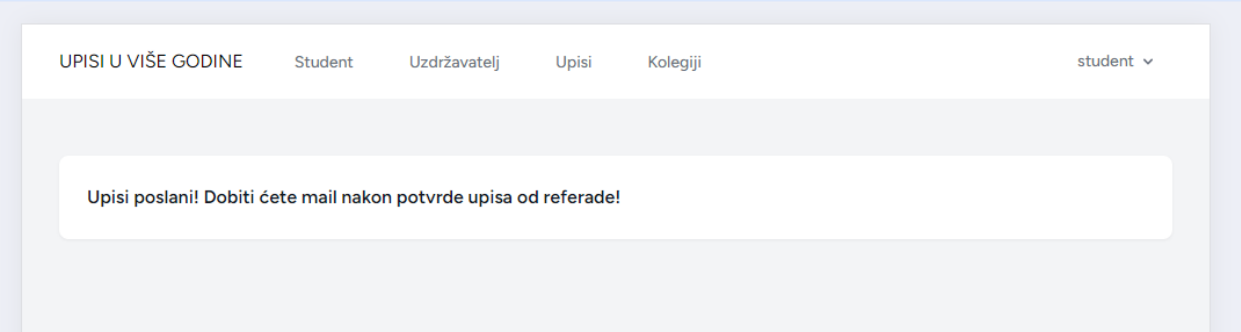
Slika 106. Prikaz forme za kreiranje upisa 1.dio



Slika 107. Prikaz forme za kreiranje upisa 2.dio

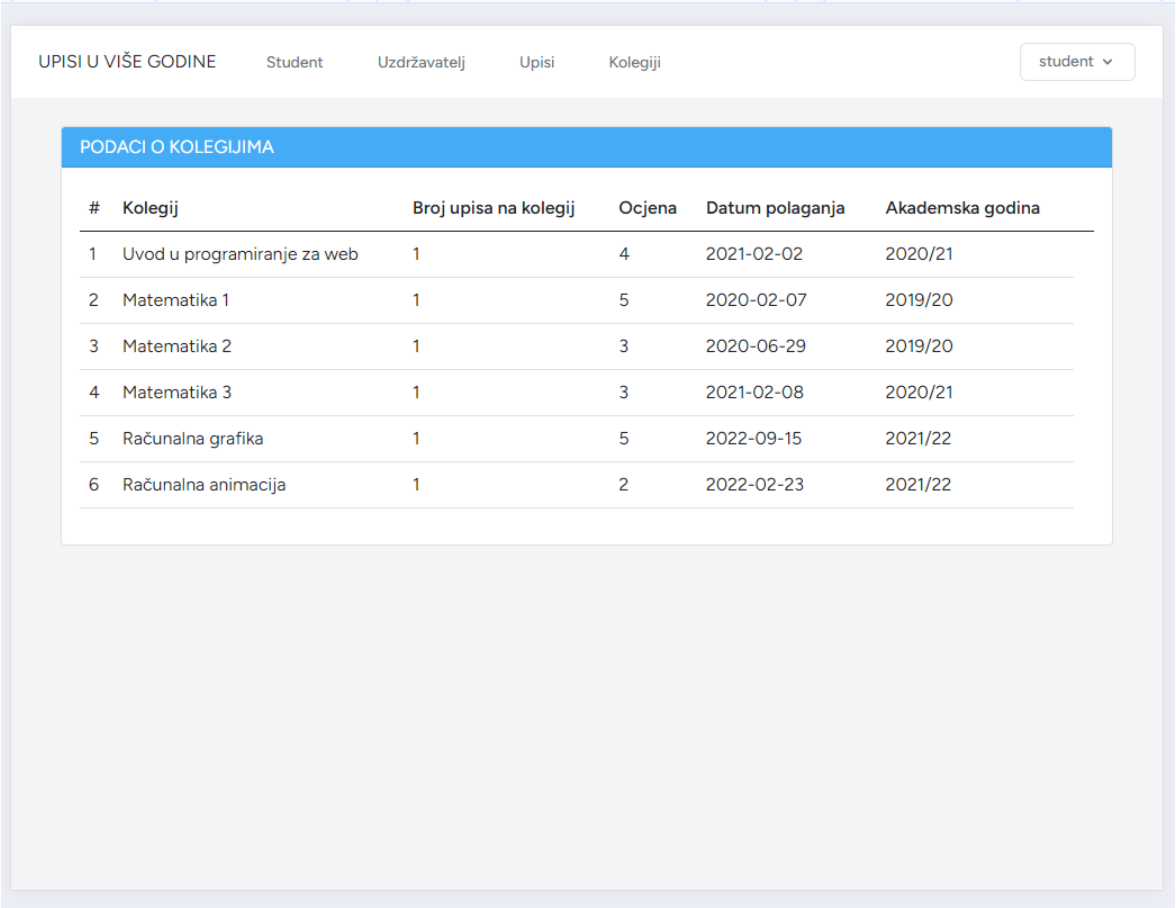


Slika 108. Prikaz liste upisa za studenta



Slika 109. Prikaz ekrana nakon klika na „Pošalji upise“

Osim upisa, student može u navigaciji kliknuti na izbornik *Kolegiji* koji otvaraju novu stranicu sa svim kolegijima studenta koje je student upisao i položio u prethodnim godina. Prikaz liste kolegija za studenta prikazana je na Slici 110.



Slika 110. Prikaz liste kolegije za studenta

7. ZAKLJUČAK

U radu je opisan proces izrade aplikacije za upise u više godina studija u laravelu. Rad započinje s opisom izrađenog modela podataka na temelju procesa upisa u više godine studija koji je i temelj za izradu aplikacije. Nakon modela podataka, opisuje se proces izrada aplikacije od izrađivanja jednostavnih tablica, tablica slabog tipa entiteta i ograničavanje pristupa i prikaza tablica s obzirom na ulogu prijavljenog korisnika. Prilikom izrade aplikacije, potrebno je obratiti pozornost na definiranje podataka u modelu te veza unutar njih kako bi se podaci kroz cijelu aplikaciju mogli dohvatiti i prikazati. Također, potrebno je obratiti pozornost i na definiranju funkcija u upravitelju i definiranju ruta kako bi se aplikacija pravilno povezala, dobila podatke, obrađivala i poslala podatke točnim funkcijama i pogledima u aplikaciji. Dobra strana laravela je fleksibilnost te definiranje funkcija na više načina i korištenjem više programskih jezika odjednom. Laravel ima veliki broj ugrađenih alata kao i biblioteka koje olakšavaju i ubrzava samu izradu aplikacije te zahvaljujući arhitekturi MVC olakšava se i održavanje aplikacije. No, laravel uz MVC arhitekturu koristi i objektno relacijsko preslikavanje za mapiranje podataka iz baze podataka u objekte programske jezike što čini laravelom vrlo složenim ukoliko se radi o većoj ili srednjoj veličini aplikacije čime se i smanjuje sama performansa.

Aplikacija za upise u više godine studija se kao i svaka druga može poboljšati, nadograditi ili proširiti sam fokus aplikacije. Ukoliko se fokus aplikacije proširi bilo bi potrebno nadograditi aplikaciju te dodati nove funkcionalnosti. Proširenje aplikacije može biti dodavanja polja za pohranu potvrde o uplaćenoj participaciji, troškovima upisa i osiguranju te automatizacija procesa stvaranja ugovora za svakog studenta. Također, jedno od proširenja može obuhvaćati implementaciju za izračun ukoliko student ima pravo upisati na pojedini kolegij s obzirom na uvjete kolegija i izračunavanje koliko puta je upisana ista razina studija. Naravno, prilikom proširenja ili nadogradnje bilo bi potrebno proširiti i sam model podataka te promijeniti logiku u kodu za pojedine funkcionalnosti.

8. LITERATURA

[1] fastfwd. 2022. “What Is Laravel? A Beginner’s Introduction.” Fastfwd. March 14, 2022. Preuzeto 19.02.2024 sa: https://www.fastfwd.com/what-is-laravel/.

[2] Surguy, Maks. 2013. “History of Laravel PHP Framework, Eloquence Emerging.” Maks Surguy’s Blog on Technology Innovation, IoT, Design and Code. July 27, 2013. Preuzeto 19.02.2024 sa: https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/.

[3] “What Is Laravel? (Definition, Uses, Features) | Built In.” n.d. Builtin.com. Preuzeto 19.02.2024 sa: https://builtin.com/software-engineering-perspectives/laravel.

[4] PHP. 2019. “PHP: What Is PHP? - Manual.” Php.net. 2019. Preuzeto 19.02.2024 https://www.php.net/manual/en/intro-whatis.php.

[5] Pavlić, M. Informacijski sustavi. Školska knjiga, Zagreb, 2011

[6] “Laravel - the PHP Framework for Web Artisans.” n.d. Laravel.com. Preuzeto 04.02.2024 sa: https://laravel.com/docs/10.x/installation.

[7] “Laravel - the PHP Framework for Web Artisans.” n.d. Laravel.com. Preuzeto 04.02.2024 sa: https://laravel.com/docs/10.x/controllers#basic-controllers.

[8] “Laravel - the PHP Framework for Web Artisans.” n.d. Laravel.com. Preuzeto 15.02.2024 sa: https://laravel.com/docs/10.x/eloquent#introduction.

9. POPIS SLIKA