Web aplikacija za vođenje kataloga diplomiranih studenata / Web application for managing the catalogue of graduate students

Kodba, Ingo

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

WEB APLIKACIJA ZA VOĐENJE KATALOGA DIPLOMIRANIH STUDENATA

Rijeka, rujan 2020.

Ingo Kodba

0069082284

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

WEB APLIKACIJA ZA VOĐENJE KATALOGA DIPLOMIRANIH STUDENATA

Mentor: doc. dr. sc. Marko Gulić

Rijeka, rujan 2020.

Ingo Kodba

0069082284

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 3. ožujka 2020.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj web aplikacija**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Ingo Kodba (0069082284)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Web aplikacija za vođenje kataloga diplomiranih studenata / Web application for managing the catalogue of graduate students**

Opis zadatka:

Razviti web aplikaciju za vođenje kataloga diplomiranih studenata. Osim funkcionalnosti upisivanja u katalog, treba implementirati funkcionalnost pametnog pretraživanja unesenih studenata na temelju prosjeka ocjena, interesa za određenu vrstu posla i ostalih važnih parametara s ciljem uspješnog uspostavljanja kontakta između poslodavaca i zainteresiranih studenata. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Laravel radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije treba koristiti Angular radni okvir i razvojnu platformu. Također, treba opisati cjelokupni razvoj web aplikacije, kao i korištenje pripadajućih tehnologija unutar aplikacije.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Ingo Kodba

Zadatak uručen pristupniku: 16. ožujka 2020.

Mentor:

Doc. dr. sc. Marko Gulić

Predsjednik povjerenstva za završni ispit:

Izv. prof. dr. sc. Kristijan Lenac

IZJAVA O SAMOSTALNOJ IZVEDBI RADA

Ovime izjavljujem da sam samostalno izradio završni rad pod vodstvom mentora doc. dr. sc. Marka Gulića.

Ingo Kodba

ZAHVALA

Zahvaljujem se Marku Guliću na susretljivosti i izvrsnom mentorstvu.

Sadržaj

1. Uvod

U ovom radu opisana je web aplikacija za vođenje kataloga diplomiranih studenata. Studenti unose svoje podatke (školovanje, radno iskustvo, tip posla koji bi htjeli raditi…)unutar kataloga u kojem administrator može pretraživati studente uz mogućnost filtriranja po određenim kriterijima radi uspješnijeg spajanja potencijalnih poslodavaca s diplomiranim studentima.

Web aplikacija sadrži korisničke funkcionalnosti registracije i prijave. Također, postoji administrator koji ima sve privilegije nad unesenim podatcima korisnika kao što su izmjena njihovih osobnih podataka. Osim ažuriranja podataka, administrator može kontaktirati svakog studenta s ciljem slanja obavijesti o potencijalnim zapošljavanjima. Kao što je i prije spomenuto, svaki diplomirani student može unijeti svoja poslovna iskustva i akademsko obrazovanje kronološkim redoslijedom. Također, student može unijeti svoje osobne podatke kao što su ime, prezime, JMBAG, kontaktni broj i kontaktna elektronička pošta.

Razmjenjivanje poruka između korisnika i administratora bi trebalo biti omogućeno samo između korisnika i administratora, nikako između dva korisnika. Kod slanja privatne poruke, automatski se šalje i elektronička pošta koja sadrži obavijesti o poslanoj poruci s poveznicom pomoću koje je moguće pročitati poruku. Elektronička pošta se šalje na primarnu kontaktnu elektroničku poštu korisnika.

Najvažnije tehnologije koje su korištene u izradi ove aplikacije su Laravel [1], Angular [2] i Bootstrap [3].

Laravel je radni okvir otvorenog koda koji se koristi u izradi poslužiteljskog (eng. back-end) i klijentskog (eng. front-end) dijela aplikacije. Podržava REST arhitekturu što ga čini idealnim za suradnju sa Angularom.

Angular je klijentska tehnologija za izradu mobilnih i računalnih aplikacija koja klijentu predstavlja sučelje za komunikaciju s poslužiteljskim dijelom aplikacije.

Bootstrap je HTML, CSS i Javascript knjižnica koja omogućuje responzivan dizajn što znači da se prikaz sadržaja prilagođava veličini ekrana. Usredotočenost prvenstveno na mobilno iskustvo.

2. Opis tehnologija

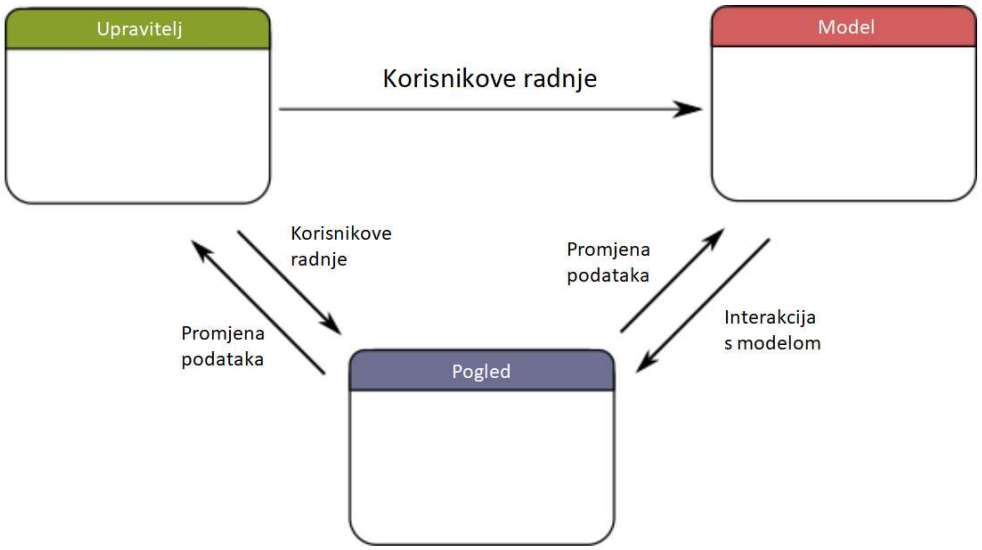
2.1. Odabir radnog okvira

Izrada web aplikacije podrazumijeva biranje prave tehnologije za ostvarenje određenog cilja. Kriteriji ove aplikacije su:

- korištenje baze podataka sa ORM tehnologijom koja omogućuje SQL upite
- gotovo, provjereno rješenje za registraciju i prijavu korisnika
- dobro dokumentiran radni okvir uz instrukcije kako ga koristiti
- otvoreni kod

2.2. Laravel

Laravel je radni okvir otvorenog pristupa (open-source) koji služi za stvaranje poslužiteljskog i klijentskog dijela web aplikacija s poslužiteljskim dijelom aplikacije baziranim na programskom jeziku PHP [4]. Laravel ima MVC (model-view-controller) arhitekturu. To je arhitektura dizajnirana tako da dijeli podatke, dizajn i logiku, respektivno, u zasebne cjeline koje međusobno komuniciraju. Upravitelj (eng. Contoller) je zadužen za funkcionalnost, model (eng. Model) je zadužen za strukturiranje podataka, a pogled (eng. View) je zadužen za dizajn stranice. Primjer MVC-a vidimo ispod na slici 2.1.



Slika 2.1. MVC arhitektura

Laravel je odabran zato što se pokazao kao jedan od najrazvijenijih radnih okvira baziranih na PHP-u i zato što zadovoljava navedene kriterije iz poglavlja 2.1.

2.2.1. Nužni uvjeti Laravela

Osnovni uvjet instaliranja Laravela je imati instaliran PHP verzije 7.3 ili više i PHP dodatke koji su navedeni u Laravelovoj dokumentaciji. Umjesto ručne instalacije, Laravelov tim olakšava i preporuča proces instalacije sa Laravel Homestead [5] virtualnim strojem.

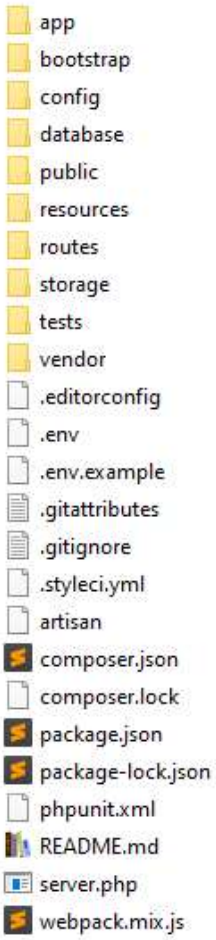
Laravel koristi alat Composer [6] za upravljanje ovisnostima. Stoga, prije instalacije, treba imati instaliran upravitelj paketa Composer. Sljedeća naredba preuzima sve potrebne ovisnosti za Laravel:

```
composer global require laravel/installer
```

Jednom kada se Laravel zajedno sa ovisnostima preuzme, pomoću sljedeće naredbe stvara se projekt tj. čista instalacija radnog okvira u odabranoj mapi, na primjer pod imenom 'blog':

```
laravel new blog
```

Struktura Laravel projekta prikazana je na slici 2.2.



Slika 2.2. Struktura Laravel projekta

U /app direktoriju nalazi se izvorni kod kao što su upravitelj i ostale .php datoteke. U /bootstrap mapi nalazi se konfiguracija za spajanje dijelova aplikacije zajedno. U /config mapi nalaze se konfiguracije za poslužitelj. U /resources mapi nalaze se pogledi i resursi potrebni klijentskom dijelu. U /database se nalazi baza podataka. U /public mapi nalaze se datoteke javno dostupne klijentima. U /routes nalaze se datoteke konfiguracije staza (eng. route). U /storage direktorij nalazi se spremište datoteka koje aplikacija koristi tijekom rada. Direktorij /tests postoji za automatizirane testne datoteke tijekom razvoja. U /vendor mapi nalaze se aplikacijske ovisnosti.

2.2.2. Javna mapa

Nakon Laravel instalacije, treba namjestiti da osnovna mapa weba bude mapa '/public'. Datoteka 'index.php' služi kao prvi kontroler za sve HTTP upite koji ulaze u aplikaciju.

2.2.3. Konfiguracijske datoteke

Sve datoteke konfiguracije su spremljene u 'config' mapi. Svaka postavka je dokumentirana u dokumentaciji dostupnoj na službenim stranicama Laravela.

Na primjer, mijenjanje naziva aplikacije (APP_NAME="Naziv aplikacije") nalazi se u .env konfiguracijskoj datoteci.

2.2.4. Aplikacijski ključ

Aplikacijski ključ je dio teksta koji se koristi za kriptiranje podataka tijekom komunikacije poslužitelja i klijenta. Poslužitelj kriptira sve kolačiće prije slanja klijentu, uključujući kolačić korisničke sjednice, i dekriptira ih kod preuzimanja koristeći taj aplikacijski ključ [7].

Sljedeće što treba učiniti je postaviti nasumičan tekst za aplikacijski ključ. Ukoliko se instalacija izvodila pomoću Composera, Composer je već postavio taj ključ. Ukoliko se instalacija radila ručno, potrebno je generirati ključ s `php artisan key:generate` naredbom.

Taj ključ se nalazi u .env datoteci. Ako aplikacijski ključ nije postavljen, komunikacija s poslužiteljem, korisničke sjednice i ostali podaci neće biti kriptirani.

2.3. **Angular**

Angular je također radni okvir otvorenog pristupa, služi za stvaranje aplikacija na klijentskoj strani. Krajnji proizvod je u HTML, CSS i Javascript sintaksi i pokreće se na klijentskom računalu.

Angular omogućuje klijentu ažuriranje prikaza web stranice u stvarnom vremenu pomoću sinkronih i asinkronih upita.

Unutar kodnog isječka 2.1. prikazan je primjer korištenja Angulara gdje se mijenja sadržaj odlomka u skladu sa unosom teksta.

Kodni isječak 2.1. Automatsko mijenjanje sadržaja odlomka

```
@Component({
  selector: 'app-loop-back',
  template: `
    <input #box (keyup)="0">
    <p>{{box.value}}</p>
  `
})
export class LoopbackComponent { }
```

Odlomak poprima vrijednost *box.value* što je zapravo tekst unesen u unos teksta iznad, označen s *#box*. S *(keyup)=* definiramo funkciju koja će se pozvati na događaj podizanja tipke tipkovnice. U ovom primjeru nije potrebno definirati događaj zato što se varijabla *box* automatski mijenja diljem cijelog programa pa i u odlomku.

2.4. Bootstrap

Bootstrap je knjižnica koja olakšava stiliziranje web aplikacije na način da izgled i funkcionalnosti koje najčešće želimo postići u web aplikacijama budu praktički standardizirani pomoću ove knjižnice.

Bootstrap je prvotno orijentiran na mobilno iskustvo, pa onda i na ostale ekrane. Razlozi zbog kojih je ovaj pristup dizajnu najbolji su:

1. Teže je pojednostaviti stolni dizajn u mobilni dizajn nego proširiti mobilni dizajn u stolni. Dizajniranje s fokusom na mobilno iskustvo, razvojni inženjeri prvo osvijeste što je najbitnije, pa onda rade dodavanja da usklade želje stolnih korisnika.
2. Pošto su mobilni ekrani manji, inženjeri i dizajneri će donijeti teške odluke odmah na početku. To im kasnije štedi vrijeme [8].

Responzivna prijelomna točka – veličina ekrana kod koje se dizajn web stranice mijenja iz jednog rasporeda u drugi. Koriste se da prikažu drugačiji raspored drugačijim veličinama ekrana. Te točke određuju granice između različitih veličina ekrana tj. koja veličina ekrana će vidjeti koji raspored.

2.4.1. Bootstrap layout

Bootstrapov raspored se sastoji od kontejnera. Kontejneri su osnovni elementi koji sadržavaju druge elemente unutar njih.

Postoje 3 klase kontejnera:

- .container – postavlja različit *max-width* kod svake prijelomne točke
- .container-fluid – uvijek je *width: 100%*, kod svih prijelomnih točaka

- .container-{naziv-prijelomne-točke} – postavi *width: 100%* do određene prijelomne točke

2.4.2. Bootstrap rešetka

Bootstrap rešetka (eng. grid) je izgrađena pomoću flexboxa. „Flexbox je prikaz rasporeda koji omogućuje elementima da poravnaju prostor unutar spremnika. Koristeći fleksibilne širine i visine, elementi se mogu poravnati tako da popunjavaju prostor ili raspodjeljuju prostor između elemenata. [9].“

Elementi se stavljaju unutar flexbox kontejnera i onda im se dodaju pravila koja ih dimenzioniraju i smještaju na mjesta. Flexbox elementi se moraju nalaziti unutar jednog od tri gore navedenih kontejnera. Na slici 2.3. prikazan je primjer rešetke,

Prvi od tri stupaca	Drugi od tri stupaca	Treći od tri stupaca

Slika 2.3. Primjer Bootstrap rešetke

U kodnom isječku 2.2. prikazan je izvorni kod za rešetku sa slike 2.1.

Kodni isječak 2.2. Primjer Bootstrap rešetke

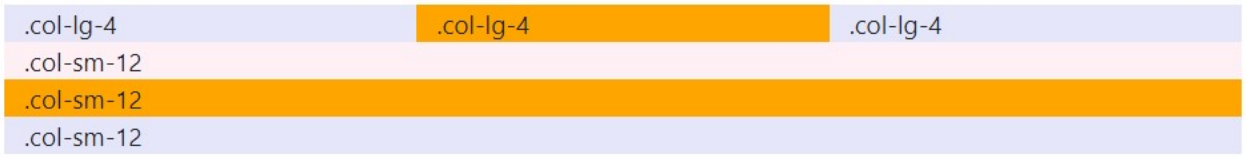
```
<div class="container">
  <div class="row">
    <div class="col">
      Prvi od tri stupaca
    </div>
    <div class="col">
      Drugi od tri stupaca
    </div>
    <div class="col">
      Treći od tri stupaca
    </div>
  </div>
</div>
```

Boostrap rešetka, koja se sastoji od redaka i stupaca, ima sljedeća svojstva:

- Klasa .row određuje jedan redak iako je pomoću trikova moguće prelomiti jedan red u više njih.
- Klasa .col određuje stupac unutar retka. Unutar kontejnera s .col klasom ubacujemo sadržaj.
- Moguće je mijenjati dužinu svakog stupca na način da ubacimo broj od 1 do 12 u col-{dužina}.
- Kod određivanja dužine stupaca vrijedi pravilo da zbroj dužina stupaca u jednom retku mora iznositi 12.

- Moguće je odrediti prijelomne točke svakom od stupaca kod kojih će se stupac iz neke njegove postavljene dužine prelomiti u potpunu dužinu kontejnera koji mu je roditelj. To se radi na način da ubacimo jednu od ključnih riječi sm, md, lg, xl nakon naziva klase .col-, ali prije dužine; col-{prijelomna-točka}-{dužina}.
- Bez dodavanja prijelomnih točaka i određenih veličina na .col elemente, ti elementi će svi imati jednake dužine, a dužina će ovisiti o tome koliko je prostora ostalo nakon ubacivanja .col elemenata sa određenom dužinom.

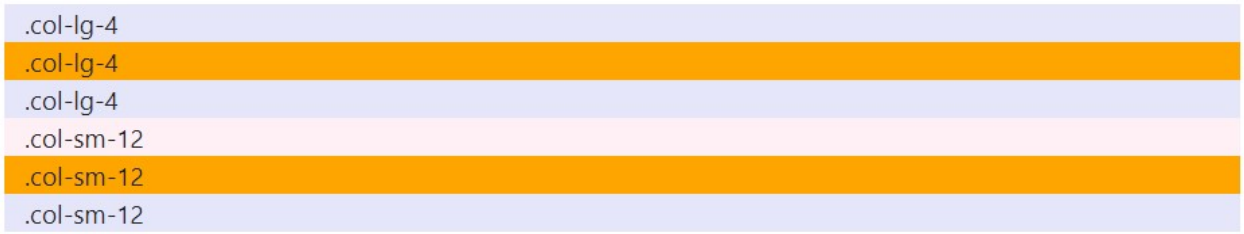
Na slikama 2.4. i 2.5. može se vidjeti primjer s dva retka. U prvom retku su tri stupca s .col-lg-4 klasom, a u drugom su tri stupca s .col-sm-12 klasom.



Slika 2.4. Prikaz bootstrap primjera na velikom ekranu

Slika 2.4. Prikaz bootstrap primjera na velikom ekranu

Drugi red se zbog svoje definicije dužine prelama u tri reda stoga imamo ukupno četiri reda. Međutim kad dođemo do prijelomne točke za manje ekrane, i prvi red se prelomi u tri retka pa imamo sveukupno šest retka (slika 2.5.).



Slika 2.5. Prikaz bootstrap primjera na malom ekranu

2.4.3. Bootstrap navigacijska traka

Bootstrapova komponenta navigacijske trake (eng. navbar) olakšava izradu responzivne navigacije te ima sljedeća svojstva:

- Navbar podrazumijeva .navbar klasu i zahtjeva .navbar-expand{-sm|-md|-lg|-xl} za responzivno urušavanje kod određenog ekrana
- Urušavanje funkcionira pomoću Bootstrapove JavaScript knjižnice
- Unutar kontejnera klase .collapse se smješta .navbar-nav, a unutar toga poveznice sa .nav-item i .nav-link klasama.

Sljedi primjer navigacijske trake koja se urušava.

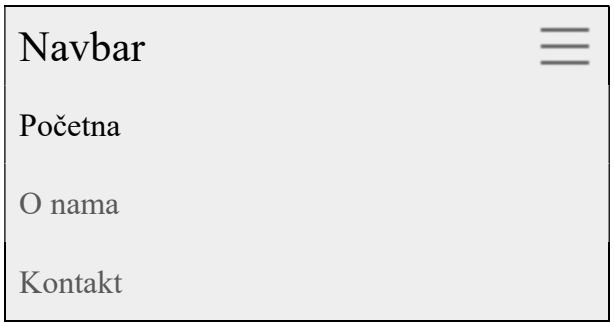
Na slici 2.6. se vidi navigacijska traka prikazana na velikom ekranu tj. na stolnom računalu.

Slika 2.6. Navigacijska traka na velikom ekranu

Kad se veličina ekrana dovoljno smanji, prijelomna točka definira kad se raspored elemenata mijenja u drugačiji raspored. Na slici 2.7. vidi se kako se navigacijska traka prelama i postaje nevidljiva sve do klika na izbornik koji se nalazi desno. To se događa kad je ekran manji od definirane prijelazne točke tj. na ekranu mobitela.

Slika 2.7. Navigacijska traka urušena na malom ekranu

Kod navigacijske trake prikazane na slici 2.7. sa desne strane se vidi gumb za proširenje navigacije. Klikom na taj gumb navigacijska traka se proširuje i izgleda kao na slici 2.8.



Slika 2.8. Navigacijska traka otvorena na mobilnom uređaju

U kodnom isječku 2.2. se može vidjeti izvorni kod takve navigacijske trake.

Kodni isječak 2.3. Primjer Boostrap navigacijske trake

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse"
    data-target="#navbarNavAltMarkup"
    aria-controls="navbarNavAltMarkup"
    aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav">
      <a class="nav-item nav-link active" href="#">Početna</a>
      <a class="nav-item nav-link" href="#">O nama</a>
      <a class="nav-item nav-link" href="#">Kontakt</a>
    </div>
  </div>
</nav>
```

2.4.4. Ostale bootstrap funkcionalnosti

Bootstrap nadjačava standardne postavke svih preglednika po vlastitim standardima. Na taj način osigurava jednak raspored i dizajn diljem različitih preglednika. Tako ima i svoje komponente kao što su uzbune (alert), značke (badge), kartice (card), karusel (carousel), kolaps (collapse), navigacijska traka (navbar), napredak (progress), vrteće animacije (spinners), tost (toast), itd.

2.5. REST

REST [10] arhitektura definira komunikaciju između poslužiteljskog i klijentskog dijela na principu jednostavnih, ali samostalno funkcionalnih upita i odgovora između poslužitelja i klijenta pomoću CRUD (create-read-update-delete) principa. U ovoj aplikaciji REST arhitektura je korištena s Angularom na primjeru 'Upravljanje državama'.

2.6. Eloquent

Upravljati bazama podataka može se s običnim SQL upitima, ali je preporučljivo koristiti Laravelov Eloquent [11]. Eloquent je ORM motor (eng. engine). "ORM je programska tehnika koja se koristi za pretvaranje podataka između dvoje ili više tipski-nekompatibilnih sustava pomoću objektno-orijentiranih programskih jezika. Ta tehnika, u suštini, čini "virtualnu bazu podataka" koju onda programski jezik može koristiti [12]". Olakšava korištenje podataka koji su u bazi podataka na način da preslika podatke u PHP objekte odnosno podaci budu strukturirani po definiranim modelima. Po MVC običaju se za ponavljajuće vrste podataka izrađuju modeli koji strukturiraju te podatke. Na primjer, na osnovu osnovnih korisničkih podatka kao npr. korisničko ime, email i lozinka, radi se model Korisnik koji definira da svaki objekt tipa Korisnik mora sadržavati ta tri podatka. Sa Eloquentom nije potrebno pisati SQL upite već se sve riješava pomoću modela i definiranih metoda.

Prilikom brisanja, u Laravelu se podaci zapravo ne brišu iz baze podataka nego se njihovo polje obrisano postavlja u logičku vrijednost istina. Na taj način redni broj novih podataka nikad neće prebrisati stare redne brojeve i redni brojevi će ostati jedinstveni, te se podaci zbog istinite logičke vrijednosti kod polja obrisano ne pojavljuju kod običnih upita, a ostaju spremljeni u memoriji u slučaju da ih želimo vratiti.

2.7. Blade

Za klijentski dio u Laravelu se koristi Blade [13] engine koji prevodi sintaksu u čisti PHP jezik, a koristi se zbog bolje snalažljivosti tijekom dizajniranja korisničkog sučelja. Bladeove dvije najbitnije odlike su nasljeđivanje predložaka i odjeljci.

2.7.1. Predlošci

Nasljeđivanje predložaka su praktički PHP-ove funkcije include i require s dodanim funkcionalnostima. Moguće je napraviti predložak koji se pojavljuje diljem više web mjesta što programerima olakšava mijenjanje i snalaženje u tom kodu.

2.7.2. Odjeljci

Odjeljci su naredbe koje govore predlošku koji dijelovi koda spadaju u koji dio predloška. Tako je moguće napraviti predložak s praznim sekcijama u koji se ubacuju funkcionalnosti. U isječku koda 2.3. se nalazi pogled koji u predložak 'osnovno' u sekciju 'sadržaj' ubacuje sadržaj 'Pozdrav svijete'.

Kodni isječak 2.4. Primjer pogleda sa sekcijom u predlošku

```
@extends('layouts.osnovno)

@section('sadrzaj')
   Pozdrav svijete
@endsection
```

Ostale funkcije Bladea su prikazivanje podataka ili kolekcija koji su pridruženi tom pogledu preko kontrolera, uvjetno grananje, korištenje čistog PHP jezika uz ljepšu otvori/zatvori sintaksu sa zaštitom od XSS (eng. cross-site scripting) i CSRF (eng. cross-site request forgery) metoda hakiranja, što su najčešće slabosti amaterski izrađenih dinamičkih web stranica. Ispod u isječku koda 2.4. se može vidjeti uvjetno grananje koje ispisuje poruke ukoliko ih ima, a ukoliko ih nema, prikazuje paragraf 'Još nema poruka'.

Kodni isječak 2.5. Primjer grananja sa *forelse* uvjetnom petljom

```
@forelse($poruke as $poruka)
   <p>{{ $poruka->text }}</p>
@empty
   <p class="text-center">Još nema poruka</p>
@endforelse
```

Kontoleri su najvažniji dio poslužiteljskog dijela upravo zato što sadrže logiku funkcionalnosti aplikacije. Oni su pozvani prilikom pokušaja dohvaćanja nekog web mjesta i zauzvrat vraćaju određeni HTTP odgovor. Najčešći odgovor je pogled popunjen potrebnim modelom i/ili ostalim podacima .

U slučaju REST aplikacije, kontroleri vraćaju tekstualne podatke koji su u obliku JSON (JavaScript Object Notation) formata. „JSON format je jedan od lakših tekstualnih otvorenih standarda dizajniran za čitljivu razmjenu podataka [14]“. Ti tekstualni podaci su većinom podaci nekog modela, ali mogu biti i multimedijskog tipa kao što su slika, zvuk i video.

2.8. Artisan

Artisan [15] je terminalno sučelje koje dolazi s Laravelom, a nudi pregršt naredbi koje olakšavaju izradu Laravel aplikacija, otklanjanje pograšaka i korištenje raznih funkcionalnosti bez potrebe programiranja. Artisanova naredba Tinker omogućuje otklanjanje pograšaka u REPL okruženju. REPL je okruženje koje koristi sljedeću petlju: pročitaj unos, evaluiraj unos, isprintaj rezultat. Pomoću te naredbe se mogu izvršiti i pregledati rezultati pojedinih dijelova koda koji bi se inače za izvršavanje morali upisali u kontroler, bez potrebe za prevođenjem. Tijekom izrade aplikacije, Tinker se najčešće koristio za dohvaćanje podataka modela i mijenjanje podataka da se vidi promjena izgleda stranice u odnosu na te podatke.

2.9. Primjer aplikacijskog zahtjeva

U ovom primjeru prikazan je proces poslužiteljeva odgovora na zahtjev. Odgovor je običan tekst 'Pozdrav svijete'.

Zahtjev koji klijent pošalje poslužitelju ima svoj životni ciklus, od primitka upita na poslužitelj do slanja odgovora natrag klijentu.

2.9.1. Routing

U web aplikacijama prva stvar koja se događa je usmjeravanje zahtjeva željenom kontroleru i njegovoj metodi. U datoteci web.php koja se nalazi u direktoriju routes upisuje se koji URL poziva koji kontroler.

U kodnom isječku 2.5. prikazan je GET zahtjev na poslužitelj kojem je URL "{naša web stranica}/hello". Taj zahtjev poziva metodu `index` iz UserController kontrolera.

Naredba jednog primjera usmjeravanja je npr. `Route::get('/hello', 'UserController@index');`

2.9.2. Odgovor kontrolera

Nakon što pomoću usmjeravanja pozovemo kontroler, kontroler je zadužen za odgovaranje na upit. Kontroler je PHP program koji se nalazi unutar nekog prostora imena (eng. namespace) i koristi određene klase za obradu zahtjeva.

Kontoler može vraćati tekst, HTML, HTTP odgovor, JSON, multimedijski objekt. Kontroler također postavlja zaglavlje, što znači postavljanje vrste odgovora, slanje kolačića, sjedničkog ključa i ostalo. U ovom primjeru vrsta odgovora je običan tekst 'Pozdrav svijete' (kodni isječak 2.6.).

Kodni isječak 2.6. Primjer odgovora upravljača

```php
<?php
   namespace App\Http\Controllers;
   use App\Http\Controllers\Controller;
   class UserController extends Controller
   {
      public function index()
      {
         return "Pozdrav svijete";
      }
   }
```

2.9.3. Primjer CRUD za model Korisnik

Laravel nudi takozvani resurs kontroler koji obavlja sve CRUD akcije s obzirom na HTTP metode i mijenjanjem URI-ja.

Da bi se omogućio CRUD za neki resurs, sve što je potrebno je u web.php upisati linija koda `Route::resource('ime_resursa', ImeKontrolera::class);`

Tada se po konvenciji tablice 2.1. može stvoriti, dohvatiti, izmijeniti ili izbrisati resurs na poslužitelju.

Tablica 2.1. Tablica konvencije za CRUD zahtjeve

Metoda	URI	Akcija	Ime staze
GET	`/photos`	index	photos.index
GET	`/photos/create`	create	photos.create
POST	`/photos`	store	photos.store
GET	`/photos/{photo}`	show	photos.show
GET	`/photos/{photo}/edit`	edit	photos.edit
PUT/PATCH	`/photos/{photo}`	update	photos.update
DELETE	`/photos/{photo}`	destroy	photos.destroy

2.10. Struktura podataka sustava

2.10.1. Korisnik

Model korisnika sadrži osnovne informacije poput imena i prezimena, dvije elektroničke pošte (jedna redundantna), JMBAG-a i skrivenog polja lozinka. Do ostalih korisničkih informacija se pristupa putem veza posredstvom ovog modela, s metodama *profile*(), *poslovi*(), *drzave*(), *skolovanja*(). Tim metodama se definiraju veze između dva ili više modela.

Moguće je definirati ove veze: jedan-prema-jedan, jedan-prema-mnogo, jedan-prema-mnogo (obrnuto), mnogo-prema-mnogo. Ovisno o tome na kojem kraju veze se nalazi predočaj, upisuje se jedna od mogućih naredbi: *hasOne*, *hasMany*, *belongsTo*, *belongsToMany*.

Primjeri veza su sljedeće:

- korisnik ima jedan profil i profil pripada jednom korisniku - jedan-prema-jedan
- korisnik ima više poslovnih iskustava, a poslovno iskustvo pripada samo jednom korisniku - jedan-prema-mnogo (isto je sa obrazovanjem)
- korisnik pripada mnogim državama u kojima je spreman raditi i države pripadaju mnogim korisnicima - mnogo-prema-mnogo

Kod modela koji imaju definirane veze možemo preko njih dobiti instance vlasnika tog modela ili djecu modela.

Korisnički račun se odvaja od korisničkog profila i ostalih informacija, odvaja nužne informacije od informativnih, da bi se tijekom korištenja aplikacije iz kontrolera u pogled proslijedili samo najpotrebniji podaci.

2.10.2. Korisnički profil

Ovaj model sadrži informacije poput dvaju brojeva mobitela (također jedan redundantan), pitanje želi li student raditi u inozemstvu (ako da, u kojim državama), koju vrstu posla preferira, tekstualno polje u koje može slobodno napisati što god želi da administrator vidi i put do slike profila.

2.10.3. Posao

Model Posao sadrži ime poslodavca ili naziv firme, naziv mogućeg posrednika, je li posao obavljan u inozemstvu, na kojem tipu posla je korisnik radio i vremenski period u kojem je korisnik obavljao taj posao.

2.10.4. Obrazovanje

Model Obrazovanje sadrži smjer, vrstu studija, prosjek ocjena i datum upisa i završetka.

2.10.5. Poruke

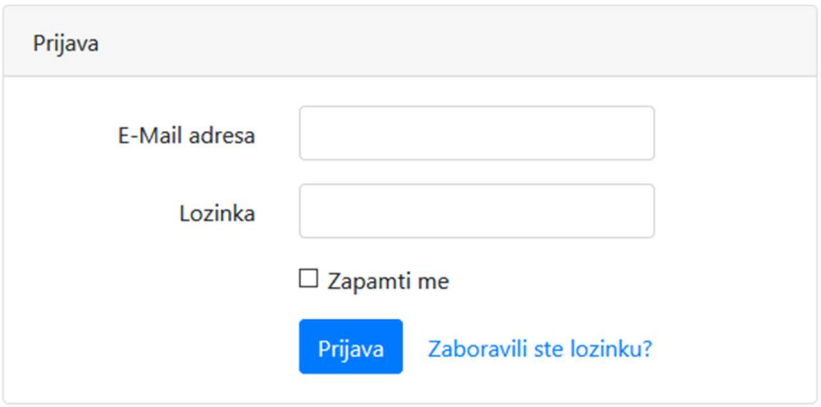
Poruke su osmišljene tako da se u bazu podataka upisuje samo redni broj korisnika koji šalje ili kojem je poruka upućena i polje binarnog tipa koje može sadržavati logičku vrijednost istina ili laž ovisno o tome je li korisnik poruku prima ili šalje. Također postoji polje koje govori je li poruka pročitana tj. otvorena.

3. Opis aplikacije

3.1. Izgled i funkcionalnost aplikacije

3.1.1. Prijava u sustav

Na slici 3.1. prikazano je sučelje za prijavu u aplikaciju kataloga diplomiranih studenata. Korisnik upisuje e-mail adresu i lozinku. Ako korisnik označi potvrdni okvir 'Zapamti me', korisnik će ostati prijavljen u sustav i nakon što napusti web mjesto. Ukoliko ne označi, odjavit će se iz sustava prilikom zatvaranja web preglednika.

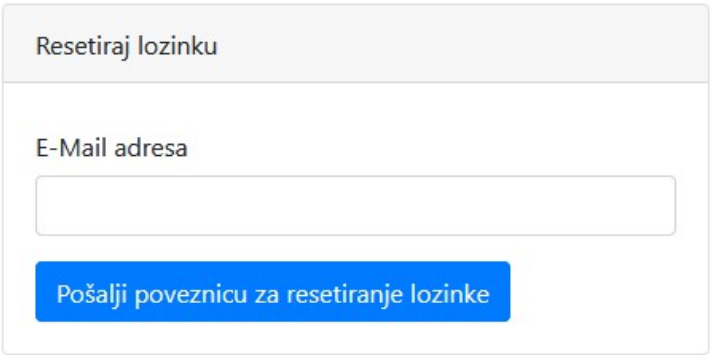


Slika 3.1. Sučelje za prijavu u sustav

Kod klika na 'Zaboravili ste lozinku?' korisnik se preusmjerava na sučelje za resetiranje lozinke.

3.1.2. Sučelje za resetiranje lozinke

Ukoliko je korisnik zaboravio lozinku, u polje unosa upisuje e-mail adresu i klikom na 'Pošalji poveznicu za resetiranje lozinke' će mu na e-mail adresu stići poveznica za resetiranje lozinke ukoliko je e-mail adresa registrirana u sustavu.



Slika 3.2. Sučelje za resetiranje lozinke

3.1.3. Korisnički profil

Nakon prijave u sustav početna stranica koja se prikazuje je korisnikov profil (slika 3.3.).

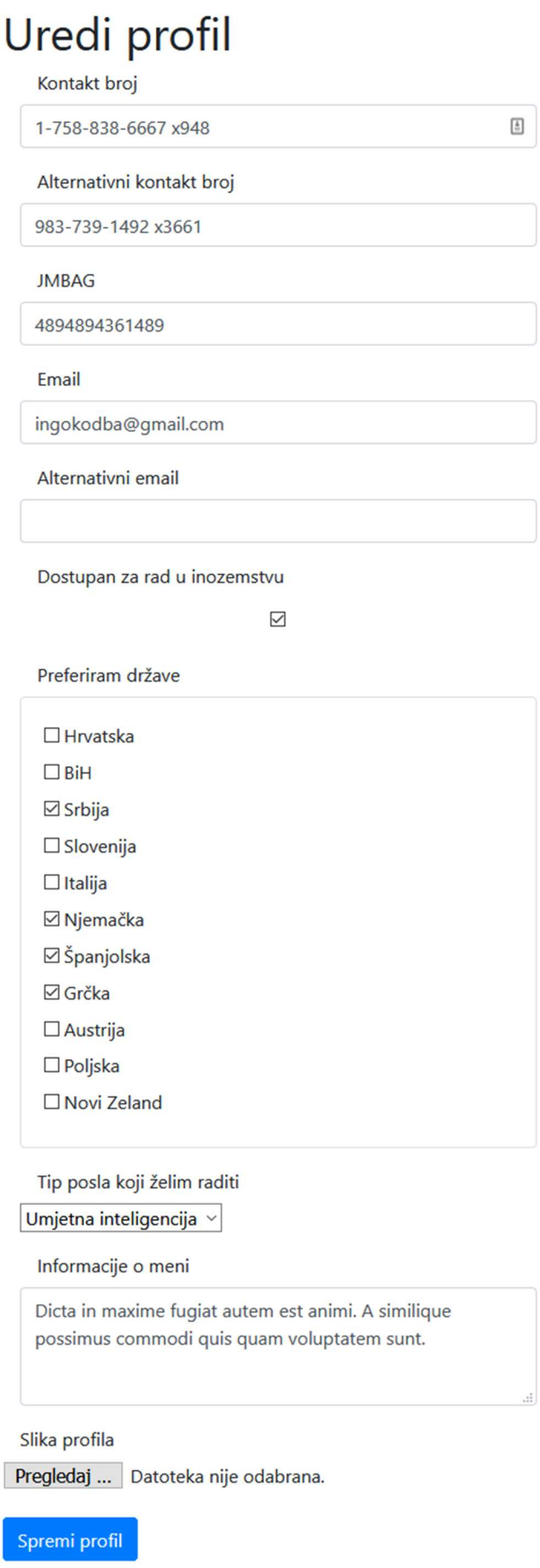
Slika 3.3. Korisnički profil

Na korisničkom profilu se u gornjoj lijevoj sekciji prikazuje slika, a u desnoj ime i prezime korisnika i njegov prilagođen opis profila.

Plavom bojom se prikazuje poveznica ‘Uredi profil’ koju vidi samo onaj korisnik čiji je profil prijavljen. Stoga ako administrator želi mijenjati korisnički profil mora se prijaviti kao taj korisnik.

U nastavku profila se nalaze informacije u kojim državama je korisnik spreman raditi, koji tip posla bi htio raditi i tri izbornika s informacijama o obrazovanju, radnom iskustvu i kontaktu. Kada se klikne na „uredi profil“ prikaže se kompletni profil korisnika koji se može ažurirati (Slika 3.4.).

Kod uređivanja profila obavezno je imati ispunjena polja ‘JMBAG’ i ‘Tip posla koji želim raditi’. Kod potvrdnoga okvira ‘Dostupan za rad u inozemstvu’ postoji funkcionalnost koja će kod uklanjanje oznake s tog okvira sakriti polje ‘Preferiram države’ pošto bi to polje bilo nepotrebno, a prikazat će se opet kod ponovnog označivanja. Ukoliko korisnik ne označi to polje, on ne želi raditi izvan Hrvatske.



Slika 3.4. Uređivanje korisničkih podataka

Postoje dva telefonska brojeva tako da se korisnika može dobiti na alternativni broj ukoliko se ne javlja preko glavnog broja. Također postoje dvije e-mail adrese ukoliko zbog bilo kojeg razloga korisnik nije u mogućnosti pristupiti glavnoj e-mail adresi ili ju ne prati. Za sliku profila moguće je, zbog sigurnosti, odabrati samo formate slika JPEG i PNG.

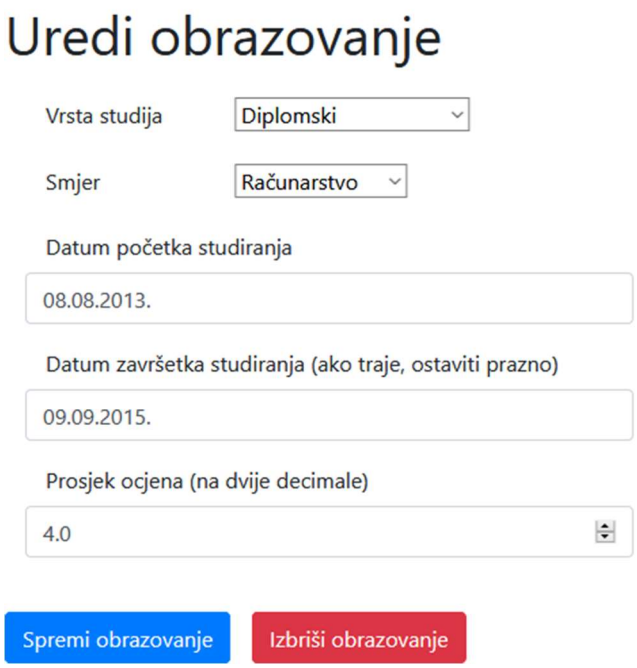
3.1.4. Obrazovanja

Na slici 3.5. prikazana su korisnikova obrazovanja i svi obrazovni podaci u njima. Moguće je dodati maksimalno tri obrazovanja.

Slika 3.5. Prikaz unesenih informacija o korisnikovom obrazovanju

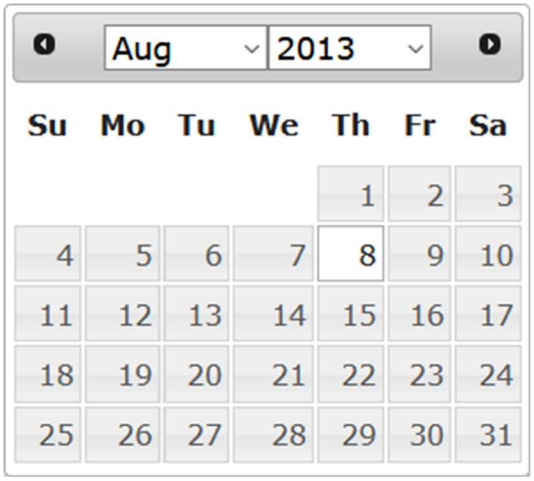
Ukoliko je uneseno tri obrazovanja, nestaje poveznica za dodavanje novog studija na dnu čija forma izgleda isto kao kod uređivanja obrazovanja (slika 3.5.).

Kod klika na poveznicu 'Uredi' korisnika preusmjerava na uređivanje tog pojedinog obrazovanja koje je prikazano slikom 3.6. U okviru 'Uredi obrazovanje' bira se vrsta studija i smjer, unose se datumi početka i završetka studiranja i procjek ocjena zaokružen na dvije decimale. Ukoliko datum završetka studiranja nije naveden, smatrat će se da korisnik još studira. Kod odabira 'Izbriši obrazovanje', studij se briše i preusmjerava se korisnika natrag na pregled obrazovanja.



Slika 3.6. Sučelje za uređivanje pojedinog studija

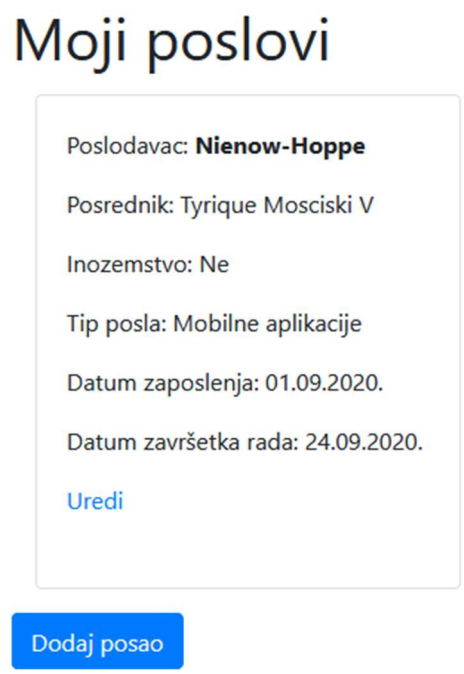
Kod odabira na bilo koje polje datuma pojavljuje se Bootstrapov okvir za odabir datuma u predefiniranom formatu (slika 3.7.). Kad se u ovom okviru odabire godina, mjesec i dan, u polje unosa će se automatski unijeti formatirani datum u formatu *dd.mm.gggg.*



Slika 3.7. Okvir za odabir datuma

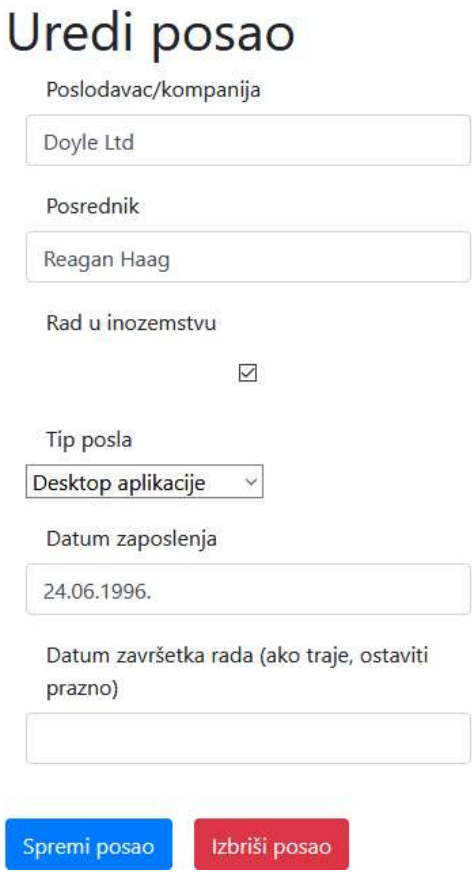
3.1.5. Radno iskustvo

Klikom na 'Moji poslovi' u navigacijskoj traci korisnik se preusmjerava na stranicu za pregled korisnikovih poslova (slika 3.8.).



Slika 3.8. Prikaz korisnikovih radnih iskustava

Klikom na 'Uredi' poveznicu kod pojedinog posla korisnika preusmjerava na sučelje za uređivanje posla (Slika 3.9.)

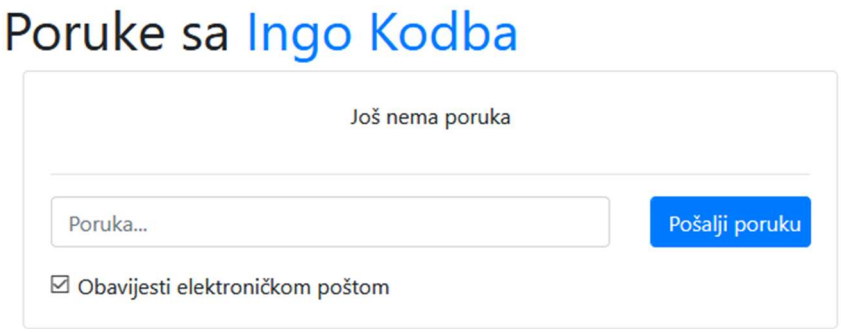


Slika 3.9. Sučelje za uređivanje posla

Ovdje se unosi ime poslodavca ili kompanije, ime posrednika (moguće je ostaviti prazno), označuje potvrdni okvir je li posao bio rađen u inozemstvu, tip posla koji je rađen i datumi početka i završetka rada. Klikom na polje unosa datuma također izbacuje okvir za odabir datuma kao na slici 3.7. Klikom na 'Izbriši posao' radno iskustvo se briše i korisnik se preusmjerava natrag na pregled poslova.

3.1.6. Dopisivanje

Na slici 3.10. prikazana je forma za dopisivanje sa korisnikom 'Ingo Kodba' iz administratorskog kuta gledišta.

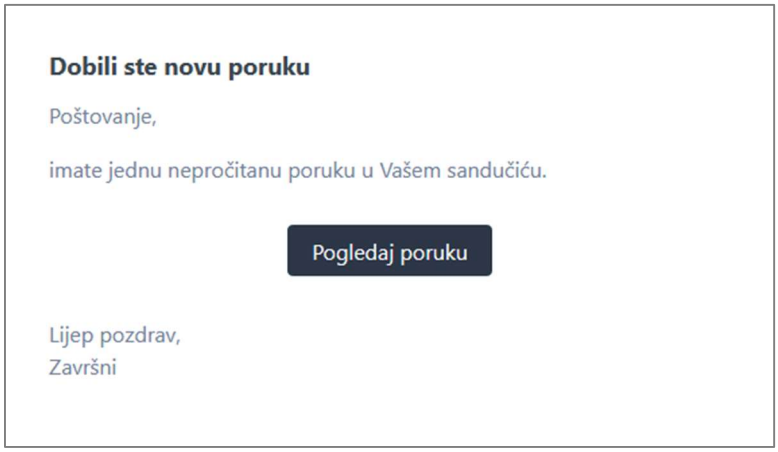


Slika 3.10. Prikaz forme za dopisivanje iz administratorskog kuta gledišta

Na formi sa slike 3.10. moguće je kliknuti na ime i prezime korisnika (tekst plave boje) za preusmjeravanje na korisnikov profil. Kod slanja poruke korisniku, korisniku se u navigacijskoj traci pojavi broj nepročitanih poruka kao što se može vidjeti na slici 3.11. Također, korisnik će dobiti obavijest elektroničkom poštom)primjer poruke prikazan je na slici 3.12.).

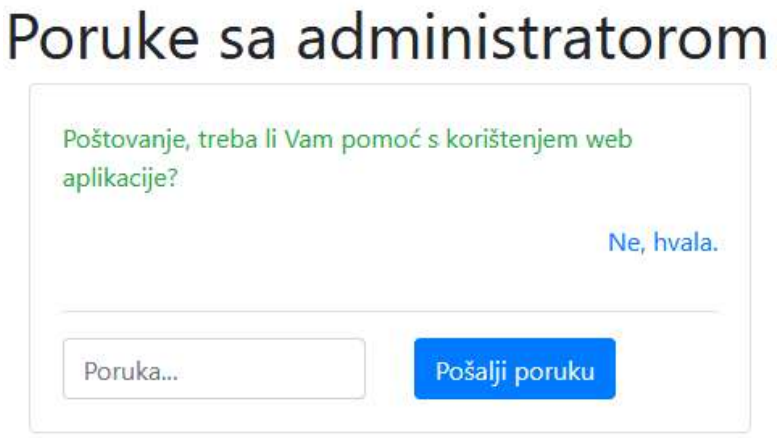
Slika 3.11. Navigacijska traka s obavijesti o nepročitanoj poruci

Kod obavijesti elektroničkom poštom ne prikazuje se poruka već samo poveznica do forme za dopisivanje u kojoj je moguće pročitati poruku (slika 3.12.).



Slika 3.12. Pristigla obavijest o novoj poruci u elektroničkoj pošti korisnika

Opcija slanja poruke elektroničkom poštom je korisna jer podsjeća korisnika na novosti koje ima u sandučiću, a pretpostavlja se kako korisnik češće provjerava elektroničku poštu nego obavijesti unutar aplikacije. Također, Možda se i korisnik zadrži unutar kataloga kako bi ažurirao svoje podatke. Razgovor s porukama između korisnika i administratora, s korisničke strane, prikazan je na slici 3.13.



Slika 3.13. Razgovor s korisničkog kuta gledišta

Na slici 3.13. prikazan je razgovor između korisnika i administratora. U ovom slučaju su poruke administratora zelene, dok su poruke korisnika plave. U slučaju da se gleda razgovor iz administratorskog kuta gledišta, administratorske poruke bi bile plave, a korisnikove zelene.

Upisivanjem teksta u unos i klikom na 'Pošalji poruku' administratoru se šalje poruka na sustav i obavijest o novoj poruci putem elektroničke pošte.

3.1.7. Angular sučelje za upravljanje državama

Na slici 3.14. se nalazi Angular klijentski dio za upravljanje državama. Ovo sučelje je izrađeno pomoću Angulara. Nakon što je implementirano u Angularu, prevodi se se u JavaScript, HTML i CSS i postavlja na poslužitelj. Stranicu je moguće prikazati klijentu zato što je u *public* mapi. Sučelje je dostupno preko administratorske navigacijske trake.

Slika 3.14. Sučelje za upravljanje državama

4. Opis funkcionalnosti aplikacije za vođenje kataloga diplomiranih studenata

4.1. Pivot tablica

Spajanje modela s vezom mnogo-prema-mnogo zahtijeva tablicu koja je posrednik između dvaju modela. Ta tablica sadrži veze između takvih modela na način da se u nju zapisuju samo primarni ključevi oba modela. Takva tablica se zove pivot tablica, a takva veza koja ostvaruje pivot tablicu nudi ove metode: *attach*, *detach*, *sync*, *syncWithoutDetaching*, *toggle*, *save* i *updateExistingPivot*. Primjer u ovoj aplikaciji koji koristi pivot tablicu je veza između korisnika i država (kodni isječak 4.1.).

Kodni isječak 4.1. Kreiranje pivot tablice pomoću Eloquenta

```
Schema::create('drzava_user', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('drzava_id');

    $table->unsignedBigInteger('user_id');
});
```

Laravelova konvencija potiče da se tablice zovu na određen način da bi bilo moguće automatsko otkrivanje veza, automatsko otkrivanje primarnih ključeva veza i ostalih prednosti Laravela i Eloquenta umjesto da se sve mora ručno navesti. U primjeru iznad tablica se zove drzava_user, dok su strani ključevi formatirani na način {model}_id.

4.2. Problem administratora i mijenjanje korisničkih podataka

Kod gotovog rješenja registracije i prijave korisnika, postoji problem jer nema različitih razina ovlasti. Da bi riješili problem ovlasti mora se uvesti administratore, vrstu korisnika zaduženu za upravljanje sustavom i mijenjanje podataka.

Postoje dva najlogičnija rješenja za ovaj problem:

- **Modelu Korisnik dodati novo polje 'admin' koje je istina ili laž vrijednosti koje odlučuje je li klijent administrator ili običan korisnik.**

  Prednosti: najjednostavnije je, za provjeru je li klijent administrator dovoljno je provjeriti je li mu to polje istina.

  Nedostaci: neiskorišteni podaci, ista stranica za prijavu, nepotrebno filtriranje korisnika kod prikaza i traženja

- **Ne dirati model Korisnik nego napraviti novi model Admin i posebnu stranicu za njegovu prijavu**

  Prednosti: moguće je u isto vrijeme biti prijavljen i kao Korisnik i kao Admin stoga je moguće kao administrator prijaviti se kao neki drugi korisnik, sigurniji pristup pošto se za administratora može napraviti sigurnost višeg stupnja, odvajanje web stranica prema različitim ulogama.

  Nedostaci: potrebno raditi stranicu za prijavu posebno samo za administratore i duplicirati protokol za autentifikaciju

Kod prvog slučaja činjenica je da model Korisnik sa sobom vuče model Korisnički profil i ostale moguće podatke običnog korisnika stoga se zauzima nepotrebna memorija.

Način da se iskoristi Laravelova funckionalnost autentifikacije sa različitim entitetima (Korisnik, Admin,...) je da se u datoteci 'config/auth.php' doda novi provider 'admins' i novi guard 'admin' (kodni isječak 4.2.).

Kodni isječak 4.2. Konfiguracija dodatne autentifikacije

```
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => App\User::class,
    ],

    'admins' => [
        'driver' => 'eloquent',
        'model' => App\Admin::class,
    ],
]
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],



    'admin' => [
        'driver' => 'session',
        'provider' => 'admins',
    ],
].
```

Sada se može mijenjanjem čuvara kod autentifikacije, umjesto obične Auth::check() provjere za obične korisnike, provjeriti je li administrator prijavljen, a to se izvršava naredbom `Auth:: guard ('admin')->check();`

Zadani administratorski podaci su:

- E-mail: admin@zavrsni.com
- Lozinka: 12345678

4.3. Konstante u aplikaciji

Podaci koji su kao što su države svijeta, vrste mogućih poslova koje se mogu izabrati, te vrste studija, su konstante koje se pojavljuju diljem web aplikacije i otprilike su stalne. Jedine konstante koje se mogu mijenjati preko običnog web administratora su države svijeta, pomoću Angulara. Za mijenjanje ostalih konstanti potrebno je urediti programski kod.

Kod dodavanja novih podataka ili mijenjanje postojećih, potrebno je ponovno pokrenuti poslužitelj ili napisati određenu zapovijed u terminal da promjena nastupi na snagu.

4.4. Autorizacija

Kad postoje korisnici s različitim privilegijama, nije dovoljno da je korisnik prijavljen u sustav. Potrebno je utvrditi ima li korisnik pravo stvarati, čitati, mijenjati ili brisati podatke. Većinom se radi o dopuštenju da korisnik mijenja vlastite podatke, dok mu je zabranjeno mijenjati podatke drugih korisnika.

Pravilo kod dizajniranja sigurnih web aplikacija je pretpostavka da su svi korisnici zlonamjerni i da će pokušati napraviti što je više štete moguće. Korisnici mogu mijenjati URL, kolačiće, sami generirati upit poslužitelju i još mnogo toga, stoga bez potrebnih autorizacija ne može se oslanjati samo na prikaz odnosno skrivanje poveznica do ranjivih mjesta.

Laravel nudi dva osnovna autorizacijska rješenja, a to su vrata i politika. Politika je vrsta autorizacije koja okružuje neki model ili resurs, dok vrata služe za općenitije stvari poput prikaza administratorskog sučelja. Iako je moguće koristiti oboje za istu stvar, po konvenciji se zbog različitih namjera razlikuju vrata od politike. Najčešće se u aplikaciji koristi mješavina, dakle i jedno i drugo.

4.5. Testiranje tijekom razvoja

Pošto se tijekom razvoja dodaju funkcionalnosti, to podrazumijeva često mijenjanje baze podataka i dodavanje puno nepotrebnih podataka u svrhu testiranja. Potrebno je nakon svakog testiranja

brisati testne podatke. Najlakše je izbrisati sve podatke i iznova popuniti tablice s novim podacima pomoću neke automatske radnje.

Slučajno, Laravel ima ugrađenu PHP knjižnicu Faker [16]. Faker knjižnica generira lažne podatke prema kategorijama, kao što su ime i prezime, telefonski broj, email, ime kompanije, broj socijalnog osiguranja, itd.

Tražeći po internetu [17], Artisan REPL se pokazao kao najbolje rješenje. Pomoću jedne naredbe u tom okruženju možemo obaviti niz akcija poput brisanje i stvaranje novih podataka pomoću knjižnice Faker. U datoteci console.php, koja se nalazi u /routes mapi, nalazi se kodni isječak 4.3. za naredbu 'newstart' zajedno s njenim podnaredbama.

Kodni isječak 4.3. Naredba 'newstart' i njene podnaredbe

```
Artisan::command('newstart', function () {
    $this->comment('Brisanje podataka iz tablica i migriranje');

    Artisan::call('migrate:fresh');

    $this->comment('Generiranje početnog korisnika');

    $user = new User;
    $user->ime = "Ingo";
    $user->prezime = "Kodba";
    $user->jmbag = "4894894361489";
    $user->email = "ingokodba@zavrsni.com";
    $user->password = Hash::make("12345678");
    $user->save();

    $this->comment('Generiranje ostalih 9 korisnika');

    factory(User::class, 9)->create();

    $this->comment('Generiranje admina');

    $admin = new App\Admin;
    $admin->ime = "Admin";
    $admin->prezime = "Kodba";
    $admin->email = "admin@zavrsni.com";
    $admin->password = Hash::make("12345678");
    $admin->save();

})->describe('Pokreni migrate:fresh i generiraj korisnike i admina');
```

Time se stvorila naredba '*newstart*' koja će na početku pozvati naredbu '*migrate:fresh*' koja će očistiti bazu podataka i obaviti migracije, a potom generirati korisnike. Nakon brisanja svih podataka iz tablica, više ne postoje korisnički profili, samo korisnički računi.

Svaki model ima svoje metode koje se pozivaju prilikom okidanja nekog događaja. Događaji su radnje prilikom kojih se obavijesti sustav da su se dogodile. Konkretno, događaji modelmodela su: *retrieved*, *creating*, *created*, *updating*, *updated*, *saving*, *saved*, *deleting*, *deleted*, *restoring*, *restored*.

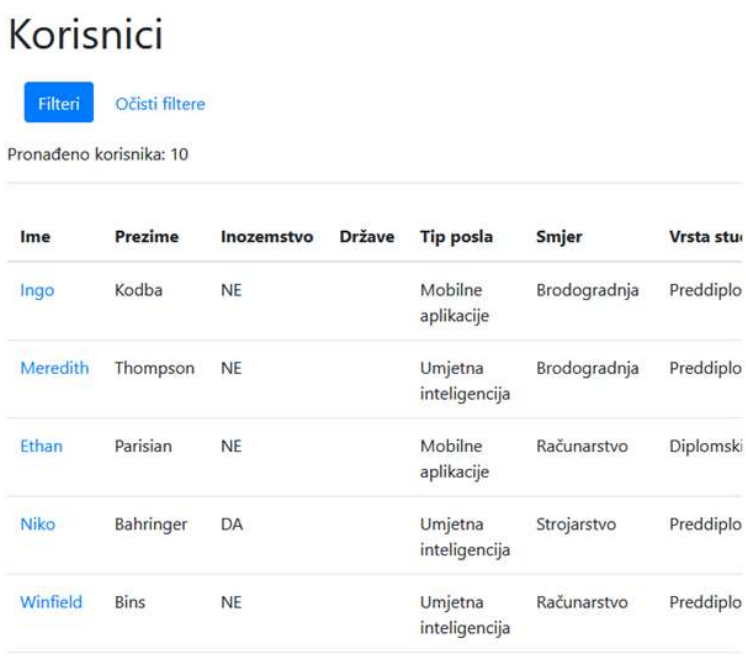
Svaki model ima mogućnost pozvati metodu *boot* prilikom stvaranja samog sebe što čini ovu metodu izvrsnom za obavljanje radnji koje se obavljaju kod registracije ili generiranje korisnika. Unutar kodnog isječka 4.4. prikazana je prazna metoda *boot*. U ovom kodnom isječku *boot* metode, prikazuje se stvaranje korisničkog profila.

Kodni isječak 4.4. Korisnikova *boot* metoda

```
protected static function boot()
{
    parent::boot();

    static::created(function ($user) {

        factory(Profile::class)->create([
            'user_id' => $user->id,
        ]);


    });
}
```

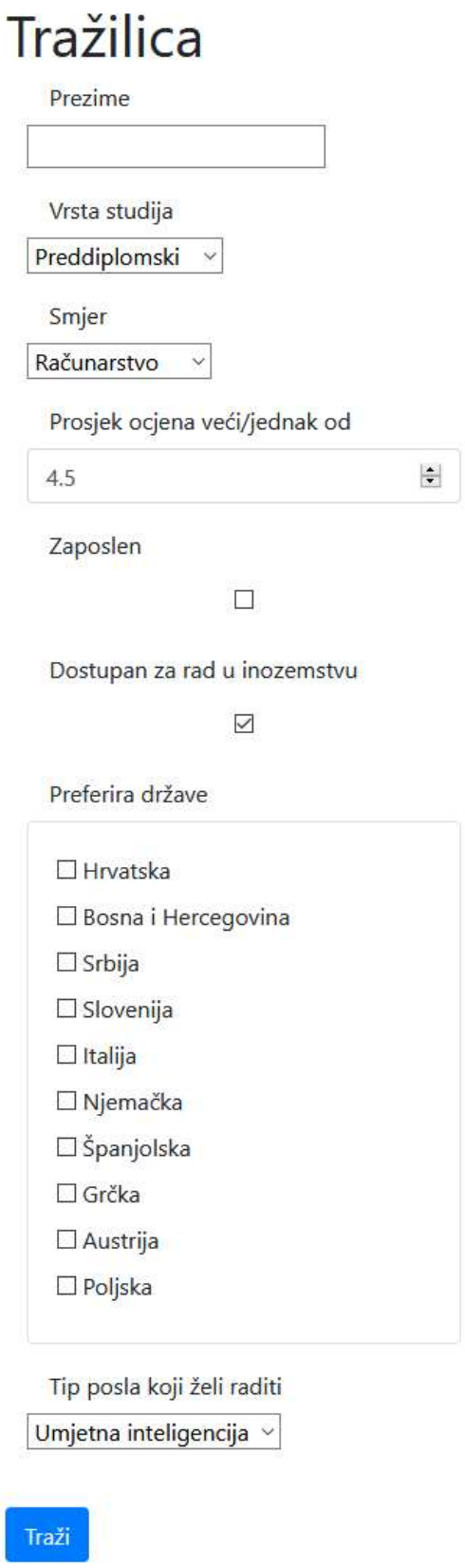
4.6. Tražilica

Kad korisnik klikne na poveznicu koja ga preusmjerava na http://127.0.0.1:8000/admin/list_users, poziva se kontroler TrazilicaController i njegova metoda *list_users* koja korisnika usmjerava prema tražilici (*Route::get('/admin/list_users', 'TrazilicaController@list_users')*;) Kod prvog posjeta na stranicu, kontroleru se šalje upit da dohvati sve korisnike sa svim njihovim podacima (slika 4.1.).



Slika 4.1 Dohvaćanje korisnika bez korištenja filtera

Međutim, kada koristimo formu tražilice prikazanu na slici 4.1., kontroleru se šalju kriteriji po kojima mora filtrirati korisnike.



Slika 4.2. Forma za filtriranje korisnika

Metoda *list_users* prima argument $request tipa Request, što je zapravo HTTP upit. Nakon zahtjeva, prvo se prikupljaju svi korisnici u kolekciji $users zajedno s njihovim profilima,

državama u kojima bi radili, obrazovanjem i poslovnim iskustvom. Naredba za dohvaćanje svih korisnika izgleda ovako: **`$users = User::all()->load('profile', 'drzave', 'skolovanja', 'poslovi');`**

Zatim se definira prazno polje *$ispadaju* u koje će se unositi svi oni korisnici koji ne zadovoljavaju kriterije filtera (naredba **`$ispadaju = array();`**)

Svakom se korisniku dodaju varijable kao što su najviši stupanj studija (preddiplomski, diplomski, poslijediplomski), smjer tog studija i prosjek (kodni isječak 4.5.).

Kodni isječak 4.5. Određivanje bitnijih korisničkih podataka

```
foreach ($users as $user)
{
  $vrsta_studija = -1;
  $smjer = -1;
  $prosjek = -1;

  $skolovanja = $user->skolovanja()->get();
  foreach ($skolovanja as $skolovanje) {
    if($skolovanje->vrsta_studija > $vrsta_studija) {
      $vrsta_studija = $skolovanje->vrsta_studija;
      $smjer = $skolovanje->smjer;
      $prosjek =$skolovanje->prosjek_ocjena;
    }
  }

  $zaposlen = false;
  $poslovi = $user->poslovi()->get();
  foreach ($poslovi as $posao) {
    if(is_null($posao->datum_zavrsetka)) {
      $zaposlen = true;
    }
  }

  $user->vrsta_studija = $vrsta_studija;
  $user->smjer = $smjer;
  $user->prosjek = $prosjek;
  $user->zaposlen = $zaposlen;
}
```

U gornjem isječku koda dohvaćaju se po svi korisnici jedan po jedan. *Foreach* petlja uzima dva argumenta: prvi argument je polje iz kojeg se izvlače druga polja, a drugi definira korisnik, upisuje ime varijable koja će postati izvučeno polje u svakoj iteraciji izvlačenja.

Unutar glavne *foreach* petlje nalaze se dvije manje *foreach* petlje koje definiraju najrelevantniji studij unutar rezultata pretrage i informaciju o korisnikovom zaposlenju, respektivno.

Neposredno prije tih dviju unutarnjih *foreach* petlji može se vidjeti da postoji dohvaćanje obrazovanja i poslovnih iskustava iz baza podataka čije varijable sadrže upravo ta polja koja se izvlače u tim petljama.

Iz funkcije *list_users* poziva se funkcija *izbaci_uljeze* koja prima argumente Request (preko vrijednosti), $users i $ispadaju (preko reference). Ta funkcija validira podatke, što znači da osigurava da podaci koji su poslani ne sadrže ništa osim teksta i vraća polje sigurnih podataka. Podaci bi inače mogli sadržavati XSS sintaksu ili ostale zlonamjerne programe.

Kodni isječak 4.6. Validiranje zahtjeva forme

```
$data = $request->validate([
  'vrsta_studija' => '',
  'smjer' => '',
  'prosjek_ocjena' => '',
  'inozemstvo' => '',
  'preferira_drzave' => '',
  ' tip_posla' => '',
]);
```

Foreach petlja u filtrira, odnosno provjerava sadrži li zadani Request polja iz forme koji bi filtrirali rezultate. Na primjer, provjerava se ako Request sadrži filter ‘smjer’. Ako sadrži, radi se filter po vrijednosti tog polja.

Kodni isječak 4.7. Primjer filtriranja rezultata po parametru forme 'smjer'

```
if (!is_null($request->input('smjer'))) {
  if ($request->input('smjer') != $user->smjer) {
    $ispada = true;
  }
}
```

Ukoliko je bilo koja provjera promijenila varijablu *$ispada* u logičku vrijednost istina, ubacuje se tog korisnika u polje *$ispadaju* koje će onda koristiti vanjska metoda, *list_users*, da ga izbaci iz prikaza rezultata (kodni isječak 4.8.).

Kodni isječak 4.8. Određivanje buduće izbačenih korisnika

```
if ($ispada) {
  array_push($ispadaju, $user);
}
```

Nakon izlaska iz unutarnje metode, algoritam se vraća na *list_users* gdje se izbacuju korisnici koji nisu zadovoljili kriterije filtriranja pomoću funkcije *diff* (**`$novi = $users->diff($ispadaju);`**). *diff* je Laravelova funkcija koja obrađuje kolekcije kao što je $users.

Metoda kod izlaza poziva sljedeću metodu *prikazi* kojoj prosljeđuje korisnike kao argument (**`return $this->prikazi($novi);`**).

U metodi *prikazi* (kodni isječak 4.9.) se samo dodaju nepromjenjive varijable poput liste smjerova, vrsta studija, tipova posla, itd. i pridružuju pogledu, pomoću metoda *'with'* i *'compact'*, koji je zadužen za prikaz rezultata.

Kodni isječak 4.9. Metoda prikazi

```
public function prikazi($users)
{
  $smjerovi = config('konstante.smjerovi');
  $vrste_studija = config('konstante.vrste_studija');
  $tipovi_posla = config('konstante.tipovi_posla');
  $smjerovi = config('konstante.smjerovi');
  $drzave = Drzava::all();

  return view('trazilica.list_users')->with(compact('smjerovi',
'vrste_studija', 'tipovi_posla', 'drzave', 'users'));
}
```

Kod izlaza iz metode *prikazi,* program vraća rezultat metodi *list_users* gdje ta metoda isto izlazi i vraća rezultat. Rezultat zadnje metode je pogled s proslijeđenim varijablama iz kontrolera (smjerovi, vrste_studija, tipovi_posla, smjerovi, drzave i users).

U pogledu 'list_users.blade.php' događa se sljedeće.

- Korisniku se, koristeći varijablu *$vrste_studija*, nude opcije po kojima može izvršiti filter (kodni isječak 4.10.).

Kodni isječak 4.10. Blade foreach petlja za odabir vrste studija

```
<select name="vrsta_studija" id="vrsta_studija">
  <option></option>

  @foreach ($vrste_studija as $key => $value)
    <option value="{{ $key }}">
      {{ $value }}
    </option>
  @endforeach

</select>
```

- U liniji koda *** Pronađeno korisnika: {{ count($users) }} *** ispisuje se broj korisnika koji se prikazuju pretraživaču:
- Zatim se ispisuje korisnik po korisnik zajedno s pripadajućim informacijama (kodni isječak 4.11.):

Kodni isječak 4.11. Ispisivanje korisnika po korisnika u Bladeu

```
@foreach($users as $user)

  <tr>
    <td><a href="/profile/{{ $user->id }}">{{$user->ime}}</a></td>
    <td>{{$user->prezime}}</td>
    <td>{{ $user->profile->inozemstvo ? 'DA' : 'NE' }}</td>
    <td>
      @foreach($user->drzave as $drzava)
        {{$drzava->naziv}},
      @endforeach
    </td>
    <td>{{ $tipovi_posla[$user->profile->tip_posla] }}</td>
    <td>{{ $smjerovi[$user->smjer] }}</td>
    <td>{{ $vrste_studija[$user->vrsta_studija] }}</td>
    <td>{{$user->prosjek}}</td>
    <td>{{ $user->zaposlen ? 'DA' : 'NE' }}</td>
  </tr>

@endforeach
```

4.7. Ažuriranje korisničkog profila

Kod ažuriranja korisničkog računa, nakon klika na gumb 'Spremi profil' šalje se upit vrste POST sa poljem '_method' postavljen na 'patch' na adresu '/profile/{id}', gdje je {id} redni broj korisnika koji sprema profil (kodni isječak 4.12.).

Kodni isječak 4.12. Parametri forme za ažuriranje korisničkog profila

```
<form action="/profile/{{ $user->id }}"
enctype="multipart/form-data" method="post">
   {{ csrf_field()
   {{ method_field('PATCH') }}
```

Naredba `Route::patch('/profile/{id}', 'ProfilesController@update')->name('profile.update');` definira rutu za ažuriranje profila

U kontroleru *ProfilesController*, argumenti $request tipa Request i $id tipa number šalju se metodi *update*. Request objekt sadrži cijeli unos iz forme, a $id varijabla je ekvivalentna rednom broju korisnika koji je upisan u URL. Unosu se pristupa preko *$request->input('ime_unosa')* gdje je *ime_unosa* naziv unosa koji je u HTML formi definiran s 'name' oznakom. U *update* metodi redoslijedom se događaju sljedeće stvari.

Korisnik se pretražuje u bazi podataka po rednom broju naredbom `$user = \App\User::findOrFail($user_id);`

Ukoliko korisnik pod tim rednim brojem ne postoji, metoda modela *findOrFail* korisniku vraća HTTP odgovor 404, tj. stranica nije pronađena.

Pomoću sigurnosne politike (`$this->authorize('update', $user->profile);`) provjerava se ima li klijent privilegiju mijenjati korisnički profil tog korisnika koji je naveden u URL-u. Ta linija koda šalje upit klasi *ProfilePolicy* metodi *update* da obradi uvjet. Uvjet je prikazan unutar kodnog isječka 4.23.

Kodni isječak 4.13. Sigurnosna politika kod ažuriranja

```
public function update(?User $user, Profile $profile)
{
   return intval(optional($user)->id) === intval($profile->user_id);
}
```

Prvi argument $user proslijeđen je automatski. Znak upitnik (ispred tipa Korisnik) u prvom argumentu je obavijest kompajleru da argument neće nužno svaki but biti proslijeđen. Na primjer, neće biti proslijeđen kad entitet tipa *Admin* pokuša promijeniti korisnički profil. U tom slučaju taj

uvjet ne bi bio zadovoljen ako ne bi postojala metoda *before* koja se poziva prije bilo koje CRUD metode u toj klasi. Metoda *before* prikazana je u kodnom isječku 4.14. u kojemse provjerava je li klijent prijavljen kao administrator.

Kodni isječak 4.14. Sigurnosna politika za administratore

```
public function before(?User $user, $ability)
{
    return Auth::guard('admin')->check();
}
```

Ukoliko neka od pozvanih metoda u *ProfilePolicy* vrati logičku vrijednost istina ili ne vrati ništa, program će nastaviti sa svojim tijekom. Ukoliko vrati logičku vrijednost neistine, program se prekida i klijentu se vraća HTTP odgovor 403, što znači zabranjen pristup.

Nadalje, podaci unosa se validiraju, što znači da prolaze kroz funkciju koja briše sve podatke koji nemaju određeni naziv i dodatno se provjerava jesu li one vrste koje bi trebali biti, na primjer unos 'image' mora biti formata jedno od formata slika – JPEG ili PNG (kodni isječak 4.15.).

Kodni isječak 4.15. Validacija podataka kod ažuriranja profila

```
$profile_data = $request->validate([
    'contact' => 'required',
    'alt_contact' => '',
    'jmbag' => 'required',
    'inozemstvo' => '',
    'preferira_drzave' => '',
    'tip_posla' => 'required',
    'description' => '',
    'image' => 'mimes:jpeg,png',
]);

$user_data = $request->validate([
    'email' => 'required|email',
    'alt_email' => 'email',
]);
```

Provjerava se je li korisnik označio polje 'inozemstvo'. Naime, potvrdni okvir (eng. checkbox) se ne šalje sa zahtjevom ukoliko nije bio označen stoga se mora ručno postaviti to polje u logičku vrijednost istina ili laž (kodni isječak 4.16.).

Kodni isječak 4.16. Logika iza potvrdnog okvira 'inozemstvo'

```
if(array_key_exists ("inozemstvo", $profile_data)) {
   $profile_data['inozemstvo'] = true;
} else {
   $profile_data['inozemstvo'] = false;
}
```

Zatim se upisuju ili brišu države koje je korisnik označio kao preferirane tako da se dohvate države preko korisnika i da se izvrše metode nad njima (odni isječak 4.17.). Metode *sync* i *detach* su *Eloquent* metode mnogo-prema-mnogo veza.

Kodni isječak 4.17. Sinkroniziranje veza Država-Korisnik

```
if($profile_data['inozemstvo']){
   $user->drzave()->sync($request->input('preferira_drzave'));
} else {
   $user->drzave()->detach();
}
```

Sync metoda uzima kao argument polje ključeva država i ažurira pivot tablicu 'drzava_user' tako da postoje samo one veze Država i Korisnika koje se nalaze u polju ključeva. *Detach* metoda raskida sve veze između Korisnika i Država, ostavljajući prazan skup preferiranih država.

Nadalje, provjerava se je li korisnik priložio datoteku pod ključem 'image' i koja se sprema u 'public/profilne/{redni broj korisnika}.{ekstenzija datoteke}'. Na kraju se uzima URL do te spremljene datoteke i dodaje se URL u polje koje će se u sljedećem koraku pridružiti glavnom polju *$profile_data* te će se spremiti profil (kodni isječak 4.18.).

Kodni isječak 4.18. Učitavanje slike profila

```
if($request->hasFile('image')){
   $extension = $request->image->extension();
   $request->file('image')->storeAs('public/profilne', $user
   ->id . "." . $extension);
   $url = Storage::disk('public')->url("profilne/" . $user
   ->id . "." . $extension);
   $imagearray = ['image' => $url];
}
```

Sada se ažurira korisnikov profil pomoću metode *update* s validiranim podacima (kodni isječak 4.19.).

Kodni isječak 4.19. Spremanje javnog puta do slike profila

```
$user->profile->update(array_merge(
  $profile_data,
  $imagearray ?? []
));
```

Oznaka duplih upitnika '??' je PHP operator koji provjerava je li varijabla lijevo od upitnika zadana, tj. da nije null. Ukoliko je zadana, vraća prvi operand (u ovom slučaju *$imagearray*), u suprotnom drugi (prazni *array*). Na kraju se spremaju podatci korisničkog računa (`$user->update($user_data);`) koji su email i alternativni email.

5. Zaključak

U ovom radu opisana je web aplikacija za vođenje kataloga diplomiranih studenata. Studenti unose svoje podatke (školovanje, radno iskustvo, tip posla koji bi htjeli raditi…) unutar kataloga u kojem administrator može pretraživati studente uz mogućnost filtriranja po određenim kriterijima radi uspješnijeg spajanja potencijalnih poslodavaca s diplomiranim studentima. Web aplikacija bi mogla biti korisna zato što bi olakšala pronalazak novih radnika s potencijalnim poslodavcima filtriranjem po kriterijima koji su poslodavcima nužni kriteriji za zapošljavanje.

U budućnosti bi se mogla dodati prijava pomoću sustava AAI@EduHr i integracija sa Studomatom da se automatizira unos podataka. Na taj način bi korisnicima skratilo vrijeme registracije u sustav i osiguralo točnost unesenih informacija pošto bi bile izvučene iz sigurnog i provjerenog sustava.

Za izradu aplikacije koristili su se radni okviri otvorenog pristupa Laravel, Angular i Bootstrap. Radni okviri najviše definiraju organiziranost aplikacije tako da poznavatelji nekog radnog okvira se mogu lako snalaziti u bilo kojoj aplikaciji napisanoj u tom radnom okviru. Zbog ovog razloga je povoljno koristiti radne okvire kod razvijanja veće aplikacije zato što se novi doprinosioci brže mogu uključiti u proces razvoja. Bez radnih okvira neki poslovi poput definiranja modela mogu biti nejasno definirani zbog neorganiziranosti aplikacije, pisanje forma za CRUD je nepotrebno ponavljajući posao koji najčešće bude kopiran uz manje izmjene, autentifikacija korisnika se piše od nule i najčešće promijene neki detalji sigurnosti zbog kojih je sustav nesiguran i ranjiv.

Radni okviri također pojednostavljuju kompleksne koncepte u svoje funkcionalnosti koje većinom razvojni programeri nikad ni nemaju potrebe detaljnije proučavati zbog dobro objašnjene dokumentacije. Također radni okviri sadrže knjižnice koje stvaraju razni doprinosioci i koje su integrirane u radni okvir tako da rijetko bude konflikata i dovoljno je upisati nekoliko naredbi za njihovu integraciju u aplikaciju.

Laravel nudi brojne kvalitetne knjižnice kao na primjer ugrađena knjižnica Auth koja obavlja autentifikaciju na otvoreni-kod provjeren način. Samo jednom naredbom u Artisanu moguće je kreirati stranice za registraciju, prijavu i zaboravljenu lozinku, model Korisnik i njegova tablica, kontrolere koji rješavaju sve autentifikacijske probleme.

Literatura

[1] Laravel LLC: „Laravel - The PHP Framework For Web Artisans“, s interneta, https://laravel.com/, pročitano 8.9.2020.

[2] Google: „Angular“, s interneta, https://angular.io/, pročitano 8.9.2020.

[3] Bootstrap tim: „Bootstrap · The most popular HTML, CSS, and JS library in the world.“, s interneta, https://getbootstrap.com/, pročitano 8.9.2020.

[4] The PHP Group: „PHP: Hypertext Preprocessor“, s interneta, https://www.php.net/, pročitano 8.9.2020.

[5] Laravel LCC, „Laravel Homestead - Laravel - The PHP Framework For Web Artisans“, s interneta, https://laravel.com/docs/8.x/homestead, pročitano 9.9.2020.

[6] Adermann N., Boggiano J.: „Composer“, s interneta, https://getcomposer.org/, pročitano 9.9.2020.

[7] Bathman, J.: „APP_KEY And You | Tighten“, s interneta, https://tighten.co/blog/app-key-and-you/, pročitano 9.9.2020.

[8] Bose, S.: „Defining Responsive Breakpoints : Best Practices | BrowserStack“, s interneta, https://www.browserstack.com/guide/responsive-design-breakpoints, pročitano 10.9.2020.

[9] „CSS flexbox“, s interneta, http://frankom.pgsri.hr/radovi/flexbox/flexbox.html, pročitano 10.9.2020.

[10] Codecademy, „What is REST? | Codecademy“, s interneta, https://www.codecademy.com/articles/what-is-rest, pročitano 10.9.2020.

[11] Laravel LLC, „Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans“, s interneta, https://laravel.com/docs/8.x/eloquent, pročitano 11.9.2020.

[12] Wikipedia: „Object-relational mapping“, s interneta, https://en.wikipedia.org/wiki/Object-relational_mapping, pročitano 11.9.2020.

[13] Laravel LLC, „Blade Templates - Laravel - The PHP Framework For Web Artisans“, s interneta, https://laravel.com/docs/8.x/blade, pročitano 12.9.2020.

[14] „JSON (format za razmenu podataka) | Web Programiranje“, s interneta, https://www.webprogramiranje.org/json/, pročitano 12.9.2020.

[15] Laravel LLC, „Artisan Console - Laravel - The PHP Framework For Web Artisans“, s interneta, https://laravel.com/docs/7.x/artisan, pročitano 12.9.2020.

[16] Zaninotto, F.: „GitHub - fzaninotto/Faker: Faker is a PHP library that generates fake data for you“, s interneta, https://github.com/fzaninotto/Faker, pročitano 12.9.2020.

[17] Repl.it, „Repl.it - The collaborative browser based IDE“, s interneta, https://repl.it/, 12.9.2020.

Popis slika

Popis kodnih isječaka

Popis kratica

JMBAG – Jedinstveni matični broj akademskog građana

ORM – Object-relational mapping

SQL - Structured Query Language

MVC – Model-View-Controller

PHP - Hypertext Preprocessor

XSS – Cross-site scripting

CSRF - Cross-site request forgery

REST - Representational state transfer

CRUD – Create, Read, Update, Delete

HTTP - Hyper Text Transfer Protocol

JSON – JavaScript simple object notation

URL - Uniform Resource Locator

URI - Uniform Resource Identifier

Sažetak

U ovom radu opisana je web aplikacija za vođenje kataloga diplomiranih studenata u svrhu pronalaska zaposlenika poslodavcima. Studentima je omogućeno unijeti podatke o svom obrazovanju, radnom iskustvu i sebi. Studenti mogu odabrati koju vrstu posla i u kojim državama preferiraju raditi tako da ih poslodavci ne kontaktiraju nepotrebno.

Neke od funkcionalnosti koje web aplikacija omogućuje su registracija i prijava korisnika, pretraživanje studenata po kriterijima i razmjena poruka. Korisnici se registriraju pomoću elektroničke pošte i lozinke. Administrator ima mogućnost prijaviti se kao korisnik i s time mijenjati korisničke podatke ukoliko se pokaže potreba. Administrator web aplikacije može pretraživati studente po vrsti studija, tipu željenog posla, prosjeku ocjena i ostalim mogućim kriterijima radi pronalaska primjerenih studenta za neki posao.

Aplikacija prvenstveno služi, kao što je i prije spomenuto, poslodavcima za pronalazak nove radne snage pomoću ovog kataloga filtriranjem studenata iz potencijalno velike baze podataka.

Tehnologije korištene u izradi ove aplikacije su Laravel, Angular i Bootstrap. Laravelove funkcionalnosti uključuju gotovo rješenje autentifikacije korisnika, a korišten je u izradi poslužiteljskog i klijentskog dijela aplikacije. Angular je korišten za izradu klijentskog dijela upravljanja preferiranim državama. Za dizajn stranica korišten je Bootstrap koji omogućuje stolno i mobilno korištenje aplikacije.

Ključne riječi: Laravel, Angular, Bootstrap, web aplikacija, student, poslodavac, katalog

Abstract

This paper describes a web application for keeping a catalog of graduate students for the purpose of finding employees for employers. Students can enter information about their education, work experience and themselves. Students can choose what type of job and in which countries they prefer to work so that employers do not contact them unnecessarily.

Some of the functionalities that the web application enables are user registration and login, searching for students according to criteria and sending messages. Users register using email and password. The administrator has the ability to log in as a user and thus change user data if the need arises. The web application administrator can search students by type of study, type of job desired, grade point average, and other possible criteria to find suitable students for the job.

The app primarily serves, as already mentioned, employers to find a new workforce using this catalog by filtering students from a potentially large database.

The technologies used in making this application are Laravel, Angular and Bootstrap. Laravel's functionalities include a ready-made user authentication solution, and was used in the development of the server and client part of the application. Angular was used to create the client part of preferred countries management. Bootstrap was used to design the site, which allows desktop and mobile use of applications.

Keywords: Laravel, Angular, Bootstrap, web application, student, employer, catalog