ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

VEB APLIKACIJA ZA POVEZIVANJE STUDENATA SA KOMPANIJAMA

– Diplomski rad–

Kandidat:

Banković Stefan 2010/345

Mentor:

doc. dr Zoran Čiča

Beograd, Septembar 2015.

SADRŽAJ

1. UVOD

Internet se identifikuje kao globalna komunikaciona mreža sačinjena od velikog broja zasebnih računara koji su povezani u jednu mrežu, takozvanu "mrežu svih mreža". Veoma je bitno da se naglasi pojam Veb-a i da se kao takav ne meša sa Internetom, pošto je česta praksa da ljudi Veb poistovećuju sa Internetom. WWW (*World Wide Web*) je najpristupačniji i najzastupljeniji Internet servis. Nastao je na osnovu idejnog projekta koji je napravio Tim Berners-Lee iz CERN-a, laboratorije za atomsku fiziku u Švajcarskoj. Tema projekta bila je sistem za hipertekst, odnosno metoda pronalaženja dokumenata na Internetu pomoću hiperveza *(hyperlink)* koje upućuju na mesta gde se dokumenti nalaze. Hiperveze se u HTML (*Hyper Text Markup Language*) dokumentima realizuju putem označavanja dela dokumenta sa navođenjem ciljnog resursa. Ove veze mogu upućivati na određeni deo dokumenta u kome se nalaze, na neki drugi dokument na istom sajtu ili na dokument koji se nalazi bilo gde na Veb-u.

Pri samom nastanku i definisanju Veb-a devedesetih godina dvadesetog veka, veb aplikacije koje su tada bile kreirane samo pomoću HTML jezika, statične veb aplikacije, zadovoljavale su tadašnje potrebe korisnika. Međutim, kako je Veb eksponencijalno rastao javila se potreba za dinamičkim kreiranjem veb aplikacija. Dinamičke veb aplikacije se ogledaju u tome da njihov sadržaj zavisi dobrim delom od korisnika i da se na osnovu podataka dobijenih od korisnika generiše HTML kod.

Dizajn veb aplikacija možemo podeliti u dva dela:

1. Deo koji se izvršava na klijentskoj strani.
2. Deo koji se izvršava na serverskoj strani.

Za izradu obe celine postoji veliki broj programskih jezika, a izbor zavisi kako od programera tako i od zahteva koji su stavljeni pred veb aplikaciju. U veb aplikaciji koja je opisana u ovom radu korišćeni su PHP (*Hypertext Preprocessor*) i MySQL (*My Structured Query Language*) na serverskoj strani i HTML, CSS (*Cascading Style Sheets*) i JavaScript na klijentskoj strani.

Ova veb aplikacija ima za cilj da poveže kompanije koje posluju na našem tržištu sa studentima i diplomcima kako Elektrotehničkog fakulteta, Univerziteta u Beogradu tako i drugih fakulteta sličnog usmerenja u Srbiji. Aplikacija je podeljena u tri dela: administrativni deo, studentski deo i deo namenjen kompanijama. Ono što je specifično za veb aplikaciju je da omogućava kompanijama da na osnovu potreba svoje kompanije pronađu optimalne kandidate. Dalje u tekstu ovog rada biće objašnjeni projektni zahtevi, zatim će biti dato upustvo za korišćenje aplikacija kao i najznačajniji delovi koda. Na kraju će biti data kratka analiza

mogućnosti unapređenja samog algoritma pronalaska optimalnog kandidata kao i koda cele aplikacije. Uz ovaj rad će biti priložen i kompletan kod sa bazom podataka.

2. PROJEKTNI ZAHTEVI

Kao što je u uvodnom delu naglašeno, u ovom delu će biti detaljno objašnjeni projektni zahtevi sa stanovišta administratora, studenata i kompanija. Osnovna ideja veb aplikacije je da sa jedne strane pomogne studentima i diplomcima u formiranju svojih radnih biografija (CV), a sa druge strane kompanijama da pronađu optimalne kandidate, kao što je već rečeno u uvodnom delu ovog rada. Naravno, krajnji cilj je potencijalno dobijanje praksi i zaposlenja mladih inženjera.

2.1. Zahtevi sa stanovišta administratora

Administrator u ovoj veb aplikaciji ima standardni zadatak kao i kod drugih veb aplikacija, a to je da vodi računa o ispravnosti podataka koje studenti, a pre svega kompanije postavljaju pomoću aplikacije. Kompanije su naglašene zato što one imaju pristup svim profilima studenata i administrator je dužan da na neki način obezbedi zaštitu privatnih podataka. Zbog toga je neophodno uvesti dodatne provere svih registrovanih kompanija pre nego im se dozvoli pristup bazi studentskih profila. Kod ovako specifičnog zadatka nije najbolje rešenje koristiti standardne provere kao što su slanje generičke poruke putem *e-mail* adrese, zadavanje prostog pitanja, jer nije potrebno otkriti da li je to čovek ili mašina ili da li je on uopšte vlasnik te adrese. Treba proveriti da li je to stvarno ta kompanija za koju se predstavlja. Iz ovih razloga, provera je zamišljena da ne bude automatska, nego da administrator kontaktira sve kompanije i direktno proveri njihove podatke.

2.2. Zahtevi sa stanovišta studenata

Studentski deo aplikacije treba da sadrži tri dela: deo na kom svaki student može videti kako izgleda njegov profil sa mogućnošću da ga u svakom trenutku može ažurirati, deo na kom mu stižu obaveštenja od kompanija i deo na kome on može gledati koje su sve kompanije registrovane. Treći deo je uveden da bi student u svakom trenutku mogao da prati koje su sve kompanije potencijalno zainteresovane za zapošljavanje novih kadrova.

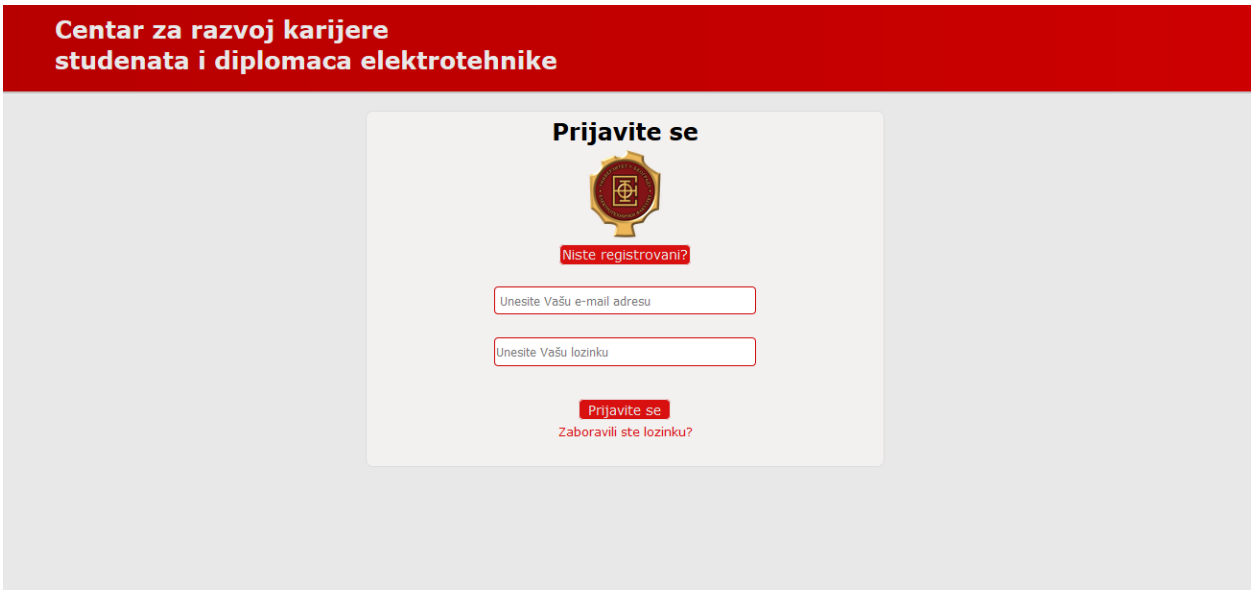
2.2. Zahtevi sa stanovišta kompanija

Deo veb aplikacije za kompanije je podeljen u dve celine: prvu na kojoj kompanije kao i studenti formiraju svoje profile koje u svakom trenutku mogu ažurirati i drugu koja služi za pretragu kandidata. Kada administrator dozvoli kompaniji pristup bazi podataka, oni mogu koristiti ovaj deo, a do tada mogu samo raditi na svom profilu. Pretraga kandidata je realizovana kroz formu u kojoj kompanija bira oblast kojom se kandidat bavi, znanja koja su im potrebna kao i obavezne i prioritetne stavke. Kada sve to izabere, aplikacija formira listu kandidata koji zadovoljavaju uslove i prikazuje kandidate od onog koji najviše odgovara potrebama kompanije do onog koji zadovoljava samo obavezne kriterijume. Kompaniji je dalje omogućeno da izabere jednog ili više kandidata i da im preko veb aplikacije pošalje obaveštenje o tome.

3. UPUSTVO ZA KORIŠĆENJE APLIKACIJE

U ovom delu rada, prvo će biti objašnjen način registracije i prijave kako administratora tako i studenata i kompanija. Zatim će redom biti dato i ilustrovano upustvo za korišćenje veb aplikacije sa stanovišta svih korisnika pojedinačno.

Sama procedura registracije i prijave se ni u čemu ne razlikuje od bilo koje druge veb aplikacije. Data je forma iz koje se preuzimaju podaci i pri registraciji se prvo proveravaju, a zatim i upisuju u korisničku bazu podataka, dok se pri prijavi samo proveravaju i zatim se vrši uspostava sesije korisnika. Početna strana veb aplikacije je zamišljena da bude strana za prijavu (slika 3.1.). Sa te strane postoji direktna veza na stranu za registraciju ako korisnik nema napravljen nalog. Takođe na slici 3.1. se može videti još jedna opcija. To je: "Zaboravili ste lozinku?". Ova opcija je uvodena kako bi se omogućilo korisnicima da u slučaju zaboravljanja šifre ipak mogu da se prijave. Naravno, postoji provera koja je u ovoj aplikaciji realizovana kroz sigurnosno pitanje, koje korisnik unosi pri ažuriranju profila. Pored pitanja, takođe unosi i odgovor pa se pri pokretanju ove opcije on proverava i ako je tačan korisniku se omogućava prijava i ažuriranje šifre.



Slika 3.1. Izgled strane za prijavu korisnika

Na strani za registrovanje (Slika 3.2.) se vrši odabir da li se korisnik registruje kao kompanija ili kao student. Te dve forme su suštinski iste, samo se razlikuju tabele u koje se upisuju studenti i kompanije. Iz tog razloga su te dve forme razdvojene. Pri dnu forme se vidi jedna veza koja

korisnike vodi do strane na kojoj su opisana pravila i odgovornosti administratora i korisnika veb aplikacije. To je neophodno kod ove aplikacije kao i kod svih drugih slične vrste zbog velike količine ličnih podataka koji se postavljaju preko iste.

Slika 3.2. Izgled strane za registraciju korisnika

3.1. Deo aplikacije namenjen administratoru

Kao što je objašnjeno u drugom poglavlju ovog rada, administrator ima zaduženje da vodi računa o podacima koje korisnici dele na aplikaciji. Stoga je njegov deo podeljen na tri strane. Prva strana je ujedno i početna na kojoj administrator ima u svakom trenutku broj registrovanih kompanija i studenata, kao i grafikon koji mu daje informacije o oblastima kojima se bave kompanije, odnosno studenti (Slika 3.3.).

Slika 3.3. Izgled grafikona za registrovane kompanije koji aplikacija formira na osnovu podataka iz baze

Druga strana služi za nadgledanje i proveru kompanija (Slika 3.4.). Ako kompaniji nije još uvek dozvoljen pristup bazi kandidata, ona je obojena u bordo boju, a ako jeste onda je samo uokvirena, kao što se može videti na slici 3.4. Takođe je omogućeno administratoru da pretražuje kompanije po oblastima rada istih. Potrebno je naglasiti da je u ovom trenutku broj kompanija koje se mogu videti na strani postavljen na pet, kako bi bila prikazana *Pagination* koja je urađena na ovoj aplikaciji. Kasnije u slučaju njenog "podizanja" na veb bi taj broj bio promenjen i postavljen na deset, ali to u ovom trenutku nije od velikog značaja. Klikom na neku od kompanija otvara se nova strana sa profilom kompanije i opcijama dozvole pristupa kompaniji ako nije dozvoljen ili brisanje kompanije iz baze (Slika 3.5.).

Treća strana služi za nadgledanje studenata i podataka koje oni postavljaju pomoću aplikacije. To je obična lista studenata koji su registrovani, a klikom na nekog od njih otvara se profil studenta sa opcijom brisanja iz baze. Kasnije kada bude bila objašnjavana pretraga kandidata biće objašnjena i ilustrovana slična lista pa iz tog razloga ona neće biti ovde ilustrovana.

Slika 3.4. Izgled strane za nadgledanje kompanija

Slika 3.5. Izgled profila kompanije sa opcijama da joj se dozvoli pristup i da se obriše iz baze

3.2. Deo aplikacije namenjen studentima

Studentski deo aplikacije je takođe podeljen u tri celine. Prva strana predstavlja profil studenta, koji se može ažurirati u svakom trenutku pritiskom na dugme “Ažuriraj profil” (Slika 3.6.).

Slika 3.6. Profil studenata

Pritiskom na “Ažuriraj profil” otvara se nova strana prikazana na slici 3.7. Odabirom oblasti rada otvaraju se dodatna polja koja su vezana za konkretnu oblast. Tako se direktno utiče na izgled profila studenta. Cela procedura formiranja radne biografije (*CV*) studenata je urađena kroz malo komplikovanu i striktnu formu iz razloga poboljšanja tačnosti pretrage kandidata. Ovo je moglo biti urađeno i na način da se podaci o studentima izvlače iz *pdf* ili *doc* dokumenata, ali to često nije najbolje rešenje zbog nepostojanja striktne forme pisanja istih. Tada bi došlo do gubitaka informacija o studentima što bi prouzrokovalo niže rangiranje istih pri pretrazi koje vrše kompanije. Na kraju, pri završavanju ažuriranja pritiskom na dugme “Ažuriraj” podaci se upisuju u bazu podataka i vraši se preusmeravanje na profil kandidata. Druga strana studenetskog dela aplikacije su obaveštenja koja kandidat potencijalno dobija od kompanija. Treća strana je skoro identična strani administratora na kojoj dobija spisak kompanija. Jedina razlika je u tome što studenti nemaju informacije kojim kompanijama je dozvoljen pristup bazi kandidata a kojima nije (Slika 3.4.).

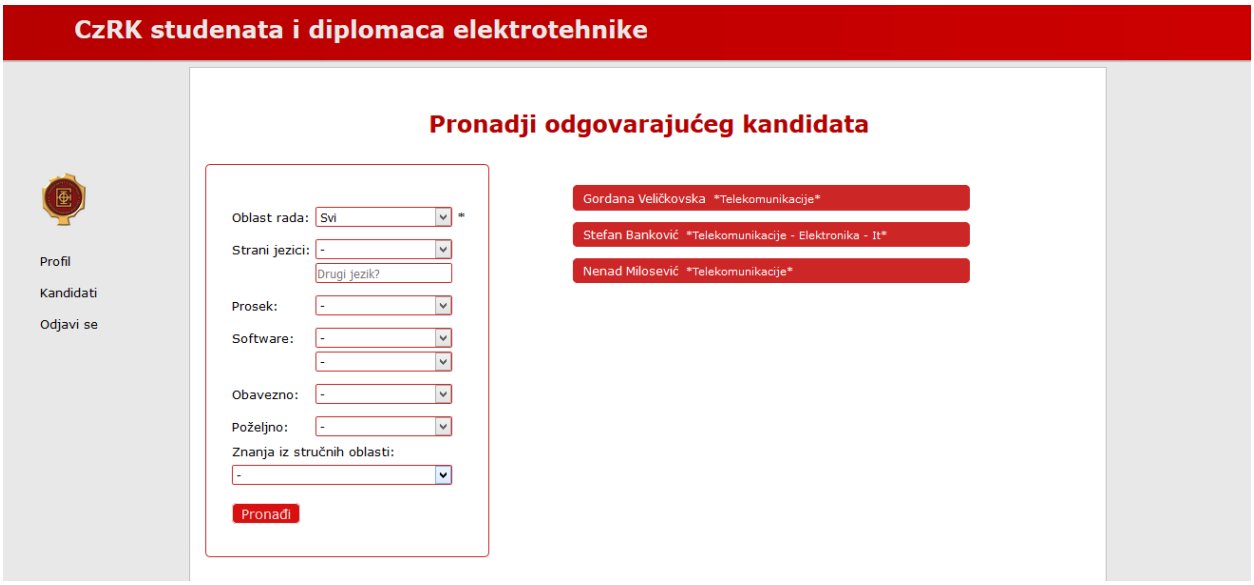
Slika 3.7. Strana za ažuriranje profila

3.3. Deo aplikacije namenjen kompanijama

Ovaj deo aplikacije je najbitniji, jer sadrži pretragu i formiranje liste potencijalnih kandidata za zaposlenje. Deo namenjen kompanijama sadrži dve celine. Prva strana je očekivana i služi za prikaz i ažuriranje profila kompanija. Ova strana koristi isti princip kao i strana koja predstavlja profil kod studenata. Pomoću dugmeta "Ažuriraj profil" preuzimaju se podaci iz baze i upisuju se u formu, zatim kompanija vrši ažuriranje i na kraju pomoću dugmeta "Ažuriraj" se vrši upisivanje svih promena podataka o kompaniji. Druga strana predstavlja jedinstvenu pretragu kandidata (Slika 3.8.). Na slici 3.8. je prikazan izgled strane sa podešenim parametrima. Ono što je veoma važno i zbog toga dodatno stoji zvezdica kod oblasti rada je da bi izvršili pretragu kompanije moraju to polje definisati. Prilikom prelaska mišem preko zvezdice ili samog dela forme namenjen oblasti rada otvara se mali prozor koji kompanije upozava upravo na to. Takođe, postoji mogućnost da se izabere obavezno polje, što bi značilo da studenti koji ne ispunjavaju taj uslov neće biti prikazani kao potencijalni kandidati. Polje "Poželjno" pruža mogućnost da se označi šta bi bilo dobro da student poseduje, ali i ako ne poseduje to znanje on može doći u obzir pri rangiranju i biće prikazan na listi. Na slici 3.9. prikazan je rezultat izabranih parametara za pretragu. Studenti su rangirani po svom znanju iz oblasti koja je izabrana kao jedini i obavezni parametar.

Slika 3.8. Izgled strane pretraga sa podešenim parametrima za pretragu

Kada se klikne na odgovarajućeg studenta veb aplikacija kopaniju vodi na njegov profil, što je gotovo identična strana prikazana na slici 3.6. Jedina razlika je u dugmićima, pa ovde postoji dugme "Pošalji obaveštenje" umesto dugmića za ažuriranje. Pri ulasku u prvi profil na listi vidi se da je student stavio za poznavanje JMS-a srednji nivo, dok su druga dva stavili početni. Što je naravno razlog ovakvog rangiranja. Pritiskom na dugme "Pošalji obaveštenje" kandidatu se šalje generička poruka kako je kompanija zainteresovana za njegovo zapošljavanje, odnosno praksu. Ta obaveštenja se naravno mogu videti na strani za obaveštenja odgovarajućeg studenta.



Slika 3.9. Izgled nakon izvršene pretrage

4. PROGRAMSKO REŠENJE

Za izradu veb aplikacije je korišćeno objektno orjentisano programiranje u PHP programskom jeziku, samim tim je kreiran određen broj klasa objekata. Pre prelaska na objašnjavanje programskog rešenja biće navedene i ukratko opisane klase koje su korišćene u izradi ove veb aplikacije.

Pri prijavi, odnosno registraciji korisnika na veb aplikaciju, koriste se klase **SesijaStudent**, **SesijaKompanije** i **SesijaAdmin**. Ove klase objekata su veoma slične, a njihova funkcija je da pomoću metoda koje su u njima definisane kreiraju sesije neophodne za čitanje svih daljih podataka o korisniku koji je prijavljen.

Na delu za administratora postoji samo jedna klasa objekata koja se koristi samo za njega. To je klasa **CitajFirmeAdmin**. Ova klasa služi za čitanje kompanija iz baze i prikazivanje na strani za administratora. Ostale klase koje se koriste su upotrebljene i u drugim delovima aplikacije. To su klase **Paginacija, ProcitajStudente** i **CitajStudent.** Prva je neophodna za kreiranje takozvane paginacije i koristi se u svim delovima aplikacije. Klasa **ProcitajStudente** namenjena je za kreiranje liste studenata, dok je poslednja neophodna za čitanje svih informacija o izabranom studentu.

Klase koje su neophodne za kompanije su: **CitanjeKompanije**, **UpisKompanije**, **UpisSlike**, **ProcitajStudente** i **CitajStudent**. **CitanjeKompanije** je klasa objekata koja ima isti zadatak sa stanovišta kompanija kao što je to **CitajStudent** za studente, a to je čitanje informacija o izabranoj kompaniji. **UpisKompanije** i **UpisSlike** kao što i ime kaže imaju zadatak da se pomoću njih vrši upis u bazu. **CitajStudent** je već opisana, a **ProcitajStudente** pored osnovnog zadatka kreiranja liste ovde ima i još jedan, a to je izvršavanje pretrage kandidata po parametrima koje kompanije zadaju.

Sa strane studenata postoji takođe nekoliko klasa. **CitajFirme** služi da napravi listu kompanija koje su registrovane na aplikaciji. Kada se izabere neka od kompanija neophodna je klasa koja će pročitati informacije o odredjenoj kompaniji. Za tu potrebu je iskorišćena već postojeća klasa **CitanjeKompanije**. Za stranu namenjenu obaveštenjima neophodna je klasa za čitanje istih i ona se zove **CitajObavestenja**. Pored ovih klasa potrebne su još klase objekata za čitanje informacija o studentima, a to su **CitajStudent** i **Student**. **CitajStudent** je već objašnjena, dok klasa objekata **Student** ima zadatak da prikaže odgovarajući sadržaj na strani za ažuriranje profila studenata.

Pošto su opisane sve kreirane klase objekata, sada će ukratko biti opisana baza podataka koja je neophodna za ovu veb aplikaciju.

Baza podataka ima osam tabela. Tabela SviStudenti je namenjena da se u nju upisuju osnovne informacije o studentu. Pored kontakt informacija, informacija o obrazovanju, u ovoj tabeli se nalaze i informacije o znanjima vezanim za jezik i *software* koje mogu posedovati studenti svih oblasti rada. U ovoj tabeli se kreira id studenta koji se dalje koristi za njegovo prepoznavanje pri čitanju podataka iz baze. Iz tog razloga u tabeli SviStudenti kolona id je podešena da bude *auto increment*. Informacije vezane za određene oblasti rada se nalaze u posebnim tabelama: Telekomunikacije, Elektronika, Energetika, Automatika i It. U ovim tabelama je podešen jedan novi id da bude *auto increment* ali on nije od nekog značaja, jer se u svaku od tabela pri upisivanju studenata upisuje id studenta koji je definisan u tabeli SviStudenti.

Kao što je opisana tabela u kojoj se nalaze informacije o studentima, tako postoji i tabela u koju se upisuju informacije o kompanijama. Za kreiranje tabele Kompanije je korišćen isti princip kao i u tabeli SviStudenti. Postoji kolona id koja je podešena da bude *auto increment* i koja služi za svako dalje prepoznavanje kompanija. Naravno, u ovoj tabeli se nalaze kolone u koje se upisuju osnovne informacije o kompanijama, istorija iste, kao i putanja do slike kompanije.

Pošto je u veb aplikaciji omogućeno da kompanije šalju obaveštenja studentima, potrebna je i tabela u koju će se ta obaveštenja upisivati. U ovoj tabeli se pri slanju obaveštenja upisuje id studenta kome se obaveštenje šalje, kao i id kompanije koja šalje to obaveštenje. Stoga su u ovoj tabeli potrebne kolone za id studenta, id kompanije kao i neke osnovne informacije o kompaniji (*e-mail* adresa, broj telefona). U ovoj tabeli postoji još i kolona za id obaveštenja koja je podešena da bude *auto increment*.

Kada su objašnjene klase objekata koje se koriste u realizaciji veb aplikacije, kao i baza podataka, može se preći na ilustrovanje samog programskog rešenja veb aplikacije.

Prvo će biti opisan proces kreiranja sesije pri registraciji odnosno prijavi, kao i sam proces registracije. Zatim će biti objašnjen deo aplikacije namenjen administratoru, pa studentima i na kraju kompanijama.

```
//Pravimo sesiju za studente
class SesijaStudent{
                public  $ime;
                public  $id;

        public function __construct($ime,$id){
                                session_start();
                                $this->ime=$ime;
                                $this->id=$id;
                                        }
        public function KreirajSesiju(){
                                $_SESSION["tip"]="student";
```

```
                                $_SESSION["id"]=$this->id;
                                $_SESSION["ime"]=$this->ime;
                                        }
        public function UnistiSesiju(){
                                session_unset();
                                        }
                        }
```

Slika 4.1. Kreiranje sesije za studenta pomoću klase SesijaStudent

Na slici 4.1. je dat izgled klase za kreiranje sesije studenta. U klasi su definisane metode za kreiranje sesije kao i za resetovanje iste, što se može i zaključiti sa slike koda. Isti princip je upotrebljen i pri kreiranju sesije administratora i kompanija, pa nije neophodno i to prikazivati. Na slici 4.2. prikazan je princip registracije studenata. Prvo se preuzimaju parametri koje korisnik unese u formu pri registraciji. Zatim se proverava da li *e-mail* adresa već postoji u bazi. Ako *e-mail* adresa ne postoji u bazi ona se upisuje, u suprotnom se korisnik preusmerava na registracija_greska.html . Kada je korisnik upisan, čita se njegov id iz baze i kreira objekat za uspostavljanje nove sesije u koju se smeštaju njegovi parametri. To se izvršava prostim pozivom odgovarajuće metode klase **SesijaStudent**. Kada je i to završeno, korisnik se preusmerava na stranu za ažuriranje profila pošto je prvi put prijavljen na veb aplikaciji. Pošto se njegov id nalazi u sesiji, pri pokretanju svake strane preuzima se i na osnovu njega se generišu informacije korisniku. Princip prijave studenata je gotovo identičan registraciji. Jedina razlika je što ne postoji upisivanje u bazu podataka već samo čitanje iz nje.

```
if(isset($_POST['regstud'])){
                $ime=$_POST['imestudenta'];
                $email=$_POST['emailstudenta'];
                $lozinka=$_POST['sifrastud'];
                $provera=0;

        //proveravamo da li email vec postoji u bazi
        $upit1 = $conn->prepare ("SELECT email FROM SviStudenti
                                  WHERE email=?");
        $upit1->bind_param("s",$email);
        $upit1->execute();
        $rezultat = $upit1->get_result();
        if ($rezultat->num_rows > 0) {
                $provera=1;
                header('Location:../registracija_greska.html');}
        if($provera==0){
        //ako email ne postoji u bazi upisujemo ga i vrsimo redirekciju
                $upit2=$conn->prepare("INSERT INTO SviStudenti
                                        (imestudenta,email,lozinka)
                                         VALUES (?, ?, ?)");
                $upit2->bind_param("sss", $ime, $email, $lozinka);
                $upit2->execute();
      //povlacimo podatke iz baze da bi kreirali sesiju za odgovarajuceg
        studenta
        $upit5=$conn->prepare("SELECT email,lozinka,id,imestudenta FROM
```

```
                                SviStudenti WHERE email=?");
        $upit5->bind_param("s", $email);
        $upit5->execute();
        $rezultat1=$upit5->get_result();
        if ($rezultat1->num_rows > 0) {
                $row=$rezultat1->fetch_assoc();
                $student=new SesijaStudent($row['imestudenta'],$row['id']);
                $student->KreirajSesiju();
                header('Location:../student/studentupdate.html');
                                                        }}}
```

Slika 4.2. Princip registracije i kreiranja sesije za studente

Na slici 4.2. je prikazan klasičan princip obrade podataka dobijenih iz forme koji je korišćen kod svih formi. Ako je pritisnuto dugme za slanje podataka, preuzimaju se parametri i kasnije se koriste.

4.1. Deo aplikacije namenjen administratoru

U ovom pasusu će biti prikazana dva zanimljiva programska rešenja vezana za administratorski deo veb aplikacije. Prvo će biti prikazan način na koji se formiraju grafikoni, a zatim će biti objašnjeno na koji način se kompanijama dozvoljava pristup. Pored ova dva problema postoji naravno još nekoliko koji su morali biti rešeni, ali ova dva su specifična jer se ne pojavljuju u drugim delovima aplikacije. Stoga će ovde biti posebno opisani.

4.1.1. Formiranje grafikona na osnovu oblasti rada studenata / kompanija

Kao što je u uvodu i navedeno, ovde će biti opisan način kreiranja grafikona na osnovu broja kompanija koje se bave određenim oblastima. Isti princip je primenjen i za studente pa neće biti prikazan i taj deo koda. Prvo što je neophodno da bi se kreirao grafikon je odnos broja kompanija koje se bave pojedinim oblastima. Ovo je realizovano čitanjem kompanija iz baze podataka, a zatim pomoću naredbe *swich* su podeljene u oblasti kojima se bave i izračunat je broj kompanija koje pripadaju pojedinačnim oblastima. Iz razloga jednostavnosti ovaj deo koda neće biti ilustrovan već će se odmah preći na ilustrovanje HTML i CSS dela koda.

```
<div id="adminkompreg"><!--Kompanije grafikon-->
     <p class="trenregnaslov">Trenutno registrovanih kompanija:
                                        <?php echo $brkomp;?> </p>
     <div id="graf">
         <div class="jedanred">
               <div class="prazan"></div>
                   <p class="pargraf">Telekomunikacije</p>
          </div>
          <div class="jedanred">
              <div class="prazan"></div>
                  <p class="pargraf">Elektronika</p>
```

```
            </div>
            <div class="jedanred">
                <div class="prazan"></div>
                    <p class="pargraf">Automatika </p>
            </div>
            <div class="jedanred">
                <div class="prazan"></div>
                <p class="pargraf">Energetika</p>
            </div>
            <div class="jedanred">
                <div class="prazan"></div>
                <p class="pargraf">It</p>
            </div>
      </div><!--Kraj graf-->

</div><!--adminkompreg kraj-->

<style type="text/css">
                .jedanred:nth-child(1) .prazan{
                   background: #C2DEFF;
                   height: <?php echo $brtelproc.'%';?>;
                                                }
                .jedanred:nth-child(2) .prazan{
                   background: #9FCAFF;
                   height: <?php echo $brelekproc.'%';?>;
                                                }
                .jedanred:nth-child(3) .prazan{
                 background: #7BB6FE;
                   height: <?php echo $brautproc.'%';?>;
                                                }
                .jedanred:nth-child(4) .prazan{
                 background: #4B9AFA;
                   height: <?php echo $brenergproc.'%';?>;
                                                }
              .jedanred{
                 width: 60px;
                 height: 100%;
                 margin: 2px 5px;
                 float: left;
                 position: relative;
                 padding-left:30px;
                           }
              .jedanred .prazan{
                 width: 100%;
                 position: absolute;
                 bottom: 0px;
                                 }
</style>
```

Slika 4.3. Kreiranje grafikona

Na početku slike 4.3. je prikazan HTML deo koda, a zatim i CSS. Ono što se može izdvojiti kao zanimljiv deo izrade grafikona a što se vidi na slici gore, je da se na visinu elementa “prazan“ direktno utiče preko php promenljivih. Visina je u stvari procenat koliko zauzima svaka oblast u ukupnom broju kompanija. Naravno, svaka od oblasti je obojena različitom bojom što je

izvršeno naredbom *background* elementa “prazan“. Takođe je podešena i širina elementa kako bi to sve izgledalo onako kako se od grafikona i očekuje. Rezultat svih ovih komandi je prikazan na slici 3.3.

4.1.2. Provera registovanih kompanija i davanje privilegije pretrage kandidata / brisanje kompanija i studenata

Kao što je već objašnjeno, da bi kompanija mogla da pristupi strani za pretragu kandidata mora dobiti dozvolu od administratora. U ovom delu rada će biti prikazan kod koji ne dozvoljava kompaniji da uđe na stranu za pretragu, kao i deo koda pomoću koga administrator daje dozvolu kompanijama da pristupaju bazi kandidata.

Pre početka svake strane veb aplikacije se vrši provera sesije, koja se sastoji od toga da se proverava da li tip sesije odgovara tipu strane, npr. ako je prijavljen administrator, tip sesije je administrator. Ako se pri proveri zaključi da nije u pitanju odgovarajući tip, vrši se preusmeravanje na stranu za prijavu. Kod strane za pretragu kandidata postoji još jedan tip provere koji je i naveden u prvom pasusu, a prikazan je na slici 4.4.

```
//Ako kompaniji nije dozvoljen pristup bazi, redirektujemo je
            if($_SESSION["provera"]==0){
                                    header('Location:kompanijaprofil.html');
                                        }
```

Slika 4.4. Provera da li je kompaniji dozvoljen pristup

Na slici se vidi da ako u sesiji promenljiva “provera” ima vrednost nula, vrši se preusmeravanje na stranu za profil. Naime, pri registrovanju kompanije podešeno je da se u bazu upisuje vrednost nula u jednu promenljivu koja služi baš za ovu proveru. Kada administrator dozvoli kompanijama pristup kandidatima ta vrednost prelazi u jedan. Ovaj postupak će biti opisan dalje u tekstu.

Na slici 3.4. se vidi izgled administratorske strane “Kompanije”. Može se zapaziti da jedna kompanija, u ovom slučaju IBM, ima različit izgled od drugih. Kao što je već objašnjeno to je zato jer njoj nije dozvoljen pristup kandidatima. Ovo se postiže kodom prikazanim na slici 4.5.

```
if($info1->provera == "1"){
            echo '<p><a href="php/kompanije_admin.php?id='
                  .$info1->id.'">'.$info1->imekompanije .'</a></p>';
                              }
else {
     echo '<p id="nijeproverena"><a href="php/kompanije_admin.php?id='
           .$info1->id .'">'.$info1->imekompanije .'</a></p>';
      }
```

Slika 4.5. Različito prikazujemo kompanije koje nemaju pristup kandidatima na strani za administratora

Različito prikazivanje omogućava id paragrafa koji se dodeljuje kompanijama koje nemaju tu dozvolu. Kada je ovo urađeno, administrator zna koje kompanije treba da proveri. Kao što je

prikazano na slici 3.5, administrator ima dugme koje služi za davanje pristupa kompanijama bazi kandidata. Kada administrator dozvoli pristup kompaniji pritiskom na dugme "Dozvoli pristup" pokreće se PHP kod koji u bazi za određenu kompaniju postavlja parametar za proveru na jedan, tako da kompanija pri sledećoj prijavi može pristupiti pretrazi kandidata.

```
//Dozvola pristupa
if(isset($_POST["dozvoli"])){
          $upit=$conn->prepare("UPDATE kompanije SET provera=? WHERE id=?"
          $upit->bind_param("ii",$i,$idkomp);
          $upit->execute();
          header("Location:../adminkompanije.html");
                              }
```

Slika 4.6. Davanje dozvole kompanijama za pretragu kandidata

Isti princip se koristi i kada se vrši brisanje kandidata, samo se umesto komande *UPDATE* koristi komanda *DELETE*. Ova komanda nam omogućava da ako administrator primeti da se neko od korisnika ne pridržava pravila korišćenja veb aplikacije može da ga obriše.

4.2. Deo aplikacije namenjen studentima

U ovom poglavlju će biti opisana realizacija formiranja radnih biografija studenata (*CV*) i način čitanja i prikazivanja obaveštenja namenjenih samo određenom studentu.

4.2.1. Formiranje CV

Formiranje CV se definitivno može svrstati u najbitnije delove ove veb aplikacije pored same pretrage kandidata. Veoma je bitno da se CV formira na striktan način, kroz veliku formu. Ova procedura kasnije omogućava dobru pretragu kandidata, kao što je ranije to i opisano. Formiranje se može podeliti u dve celine. Prva celina je povlačenje podataka iz baze za određenog korisnika (u ovom slučaju to je student, ali isti princip važi i za kompanije), dok je druga logično, prikupljanje podataka iz forme i njihovo ažuriranje u bazi. Pri uspešnom registrovanju vrši se odmah preusmeravanje na stranu za ažuriranje profila i povlače se osnovni podaci iz baze: ime, lozinka i *e-mail* adresa korisnika. Izgled forme je prikazan na slici 3.7. Pritiskom na neku od oblasti kandidatu se otvaraju nova polja koja treba da popuni. To je realizovano iz razloga da se ne moraju zamarati kandidati koji se bave npr. energetikom pitanjima iz telekomunikacija. Kod ovog rešenja je realizovan u *JavaScript* programskom jeziku. Zbog formata pisanog koda koji je dosta različit od A4 formata, ovaj deo koda neće biti prikazan u radu, ali kao što je već naglašeno uz sam rad je priložen i kompletan kod pa se uvek može pregledati ceo. Ovaj deo je realizovan u dokumentu: student_update.php.

Sada će biti objašnjen princip čitanja podataka iz baze, kao i njihovo prikazivanje na strani za ažuriranje. Čitanje iz baze je u ovom slučaju realizovano kroz jednu klasu koja služi samo za to (Slika 4.7.). Na slici je pored metode za čitanje podataka iz tabele SviStudenti prikazana

```
class CitajStudent{
            public $id;
            public $conn;

            public function __construct($id,$conn){
                        $this->id=$id;
                        $this->conn=$conn;
                                                }
            public function ProcitajSviStudenti (){
                        $konekcija=$this->conn;
                        $id=$this->id;
                        $upit=$konekcija->prepare("SELECT *
                                                FROM SviStudenti
                                                WHERE id=?");
                        $upit->bind_param('i', $id);
                        $upit->execute();
                        $rezultat = $upit->get_result();
                        $info = $rezultat->fetch_assoc();
                        return $info;
                                                }
            public function ProcitajTelekomunikacije(){
                        $konekcija=$this->conn;
                        $id=$this->id;
                        $upit=$konekcija->prepare("SELECT *
                                                FROM Telekomunikacije
                                                WHERE idstudenta=?");
                        $upit->bind_param('i', $id);
                        $upit->execute();
                        $rezultat = $upit->get_result();
                        $info = $rezultat->fetch_assoc();
                        return $info;
```

Slika 4.7. Klasa za čitanje podataka za studente

i metoda za čitanje iz tabele Telekomunikacije. Za sve ostale oblasti rada metoda je identična metodi za telekomunikacije, razlika je jedino u tabeli iz koje čitamo informacije. Sada kada postoji metoda klase, dovoljno je samo da se kreira objekat klase **CitajStudent** i da se pozove odgovarajuća metoda, što je i urađeno u dokumentu studentupdate.html. Način realizacije kreiranja objekata je prikazan na slici 4.8. Kad postoje sve informacije iz baze podataka o studentu može se formirati kako forma za ažuriranje tako i njegov profil ako je to potrebno. Trenutno se posmatra forma pa će stoga biti ilustrovan način prikazivanja podataka u formi koji su prethodno izvučeni iz baze. Ovo će biti prikazano na jednostavnom primeru opcije u kojoj se vrši selekcija da ili ne na pitanje da li student poseduje vozačku dozvolu (Slika 4.9.).

```
//kreiramo objekat za citanje
$student= new CitajStudent($id,$conn);
//citamo osnovne informacije i smestam u array $osn
```

```
$osn=$student->ProcitajSviStudenti();
//na osnovu inf o oblasti studenta citamo iz ostalih tabela

$tel=NULL;$elek=NULL;$energ=NULL;$aut=NULL;$it=NULL;//predefinisane vrednosti

//Ako je u tabeli SviStudenti setovana neka od oblasti citamo informacije iz
te tabele
if($osn["telekomunikacije"]=="Telekomunikacije"){
                    $tel=$student->ProcitajTelekomunikacije();}
if($osn["elektronika"]=="Elektronika"){
                    $elek=$student->ProcitajElektronika();}
if($osn["energetika"]=="Energetika"){
                    $energ=$student->ProcitajEnergetika();}
if($osn["automatika"]=="Automatika"){
                    $aut=$student->ProcitajAutomatika();}
if($osn["it"]=="It"){
                    $it=$student->ProcitajIt();}
```

Slika 4.8. Način na koji kreiramo objekat za čitanje iz baze

Kao što se na slici 4.9. vidi, na osnovu podataka koji su dobijeni iz baze, prikazuje se HTML kod na strani korisnika. Ako je student već rekao da ima dozvolu, sledeći put kad pritisne dugme za ažuriranje profila biće mu to i podešeno. To je princip po kome se podešavaju i svi ostali podaci studenta odnosno kompanija pri ažuriranju, pa nije potrebno za svaki dodatno pojašnjavati.

```
<div id="vozacki"><!--informacije o vozackoj dozvoli-->
    <p>Da li imate položen vozački za B kategoriju?
       <select name="vozacka">
       <?php
            if($osn["vozackadozvola"]==1){
                        echo '<option value="1"> Da </option>
                              <option value="0"> Ne </option>';}
            else {
                  echo '<option value="1"> Da </option>
                        <option value="0" selected> Ne </option>';}
       ?>
       </select></p>
</div> <!--kraj informacija o vozackoj dozvoli-->
```

Slika 4.9. Primer kako pomoću vrednosti iz baze utičemo na izgled html koda

4.2.2. Obaveštenja

U ovom pasusu će biti objašnjeno čitanje obaveštenja iz baze kao i njihov prikaz na strani namenjenoj za obaveštavanje studenata. Takođe će biti iskorišćena prilika da se objasni tzv. "paginacija", odnosno podešavanje da se na jednoj strani prikazuje po deset obaveštenja, a ako ih ima više od deset ona budu raspoređena na više strana. Čitanje obaveštenja je realizovano kroz klasu koja se zove **CitajObavestenja**, dok je za kreiranje "paginacije" potrebna klasa **Paginacija**. Prvo će biti objašnjena klasa **Paginacija**, zato što se ona direktno koristi u klasi

CitajObavestenja. Ova klasa prvenstveno služi da kaže odakle treba početi čitanje obaveštenja iz baze podataka.

```
//Klasa za paginaciju
class Paginacija{
                public $trenutna_strana;
                public $brpostrani;
                public $ukupno;

                public function __construct($strana=1,$brpostrani=5,$count){
                                $this->trenutna_strana=(int)$strana;
                                $this->brpostrani=(int)$brpostrani;
                                $this->ukupno=(int)$count;
                                                                        }
                public function BrojStrana(){
                                return ceil($this->ukupno/$this->brpostrani);
                                        }
                public function Offset(){
                                return ($this->trenutna_strana-1)*
                                        $this->brpostrani;
                                       }

                public function PrethodnaStrana(){
                                 return ($this->trenutna_strana-1);
                                          }
                public function SledecaStrana(){
                                 return ($this->trenutna_strana+1);
                                          }
        }
```

Slika 4.10. Klasa za paginaciju

Npr. ako se korisnik nalazi na strani dva, a podešeno je da bude po deset obaveštenja na strani, pomoću metode *Offset* iz klase **Paginacija** će biti dobijeno da treba početi od obaveštenja broj 11 u bazi, tj. *offset* je deset. Isto tako vraća i broj strana koje je neophodno kreirati da bi bila pročitana sva obaveštenja.

Sada će biti objašnjen način realizacije čitanja iz baze pomoću parametara koje dobijamo iz klase **Paginacija**. Prvo što je neophodno da bi se realizovalo sve što je zamisliljeno je ukupan broj obaveštenja, što se dobija kao što je prikazano na slici 4.11. pomoću metode *BrojObavestenja*. Ta metoda je realizovana preko upita "*SELECT COUNT(*) AS 'Student' FROM Obavestenja WHERE idstudenta=?*" i njen zadatak je da prosto izbroji obaveštenja koja su namenjena studentu, pa ovde neće biti dodatno objašnjena. Dalje se kreira objekat klase **Paginacija** i izračunava se *offset* pomoću kog se čitaju informacije iz baze, kao što je i ilustrovano na slici. Ostali deo koda je klasično čitanje iz baze i nije ga neophodno dodatno pojašnjavati.

Sada je na red došla i realizacija same "paginacije". Ona je ilustrovana na slici 4.12. Kao što se na slici i vidi, realizovana je kroz jednu *for* petlju u kojoj se ispisuje broj strana koje postoje.

Ono što je ključno, je da se pravilno izračuna ukupan broj obaveštenja, podeli po stranama i tako raspoređuje *offset* za čitanje podataka.

```
class CitanjeObavestenja{
            public $conn;
            public $id;
            function __construct($conn,$id){
                        $this->conn=$conn;
                        $this->id=$id;
                                                }
            //Funkcija za citanje obavestenja
            public function ProcitajObavestenja(){
                        $konekcija=$this->conn;//Pokazivac na konekciju
                        $count=$this->BrojObavestenja($konekcija,$this->id);
                        $brpostrani=5;
                        //Broj strane
                        $strana=!empty($_GET["strana"])?
                                    (int)$_GET["strana"]:1;
                        $paginacija= new Paginacija
                                    ($strana,$brpostrani,$count);
                        $offset=$paginacija->Offset();

            //Citanje iz baze po odredjenim parametrima za paginaciju
            $upit=$konekcija->prepare("SELECT
                                    imekompanije,mailkompanije,
                                    telefonkompanije
                                    FROM Obavestenja
                                    WHERE idstudenta=?
                                    ORDER BY id DESC
                                    LIMIT $brpostrani
                                    OFFSET $offset");
            $upit->bind_param("i",$this->id);
            $upit->execute();
            $rezultat=$upit->get_result();
            if($rezultat->num_rows>0){
                        while($info = $rezultat->fetch_assoc()){
                              echo '<p> Kompanija <b>'.$info["imekompanije"]
                                  .'</b> je zainteresovana za Vas kao
                                   kandidata, molimo Vas javite
                                   se putem mail-a: <b>'
                                  .$info["mailkompanije"]
                                  .'</b> ili putem telefona: <b>'
                                  .$info["telefonkompanije"]
                                  . '</b> .' ;
                                      }
```

Slika 4.11. Način realizacije čitanja obaveštenja

```
    //Paginacija
    if($paginacija->BrojStrana()>1){
                echo '<p class="pag">';
                for($i=1; $i <= $paginacija->BrojStrana(); $i++){
                    if($i==$strana){
                        echo '<span class="paginacija"><b>'.$i.'</b></span>';
                                    }
                    else{
```

```
                        echo '<span class="paginacija"><a
                              href="studentobavestenja.html?strana='
                             .$i.'">'.$i.'</a></span>';
                         }
                                    }
                    echo '</p>';

    }}//Kraj if-a ako postoji citanje iz baze
else{
      echo '<b>Trenutno nemamo obaveštenja za Vas.</b>';
      }
}
```

Slika 4.12. Realizacija "paginacije"

4.3. Deo aplikacije namenjen kompanijama

Deo aplikacije namenjen kompanijama predstavlja samu srž programa jer je realizovana jedinstvena pretraga kandidata. Programsko rešenje pretrage će biti detaljno opisano u ovom poglavlju.

4.3.1. Pretraga kandidata i slanje obaveštenja

Sama pretraga je podeljena u dve glavne celine. Prvu celinu čini forma i prikupljanje parametara po kojima će kandidati biti pretraživani, dok drugu čini čitanje kandidata iz baze, formiranje liste za ispis kandidata i njihov prikaz kompaniji koja je izvršila pretragu. Na nekoliko mesta u ovom radu bilo je objašnjeno pravljenje forme i prikupljanje podataka iz istih tako da će se ovde odmah preći na drugu celinu pretrage kandidata.

Prvo, da bi se studenti uopšte mogli na neki način rangirati, mora se napraviti algoritam po kojem će oni biti ocenjivani. U ovoj veb aplikaciji je to urađeno na način da svako znanje koje kompanija zahteva od kandidata, donosi određen broj bodova svakom kandidatu koji ga poseduje. Ako kandidat iz oblasti koju kompanija zahteva ima postavljeno napredno znanje dobija četiri boda, ako ima srednje znanje dobija tri, a za početno dobija dva. Ako kandidat nije postavio nivo njegovog znanja već je samo naveo da poznaje tu oblast dobija jedan bod. Veoma je bitno naglasiti da ako kompanija postavi neku oblast na obavezno, a kandidat nema to znanje on direktno ispada sa liste i neće biti prikazan kompaniji. Ako kandidat ima traženo znanje dobija dodatne bodove po istom algoritmu kao što je prethodno opisano, što dovodi do toga da ako za obavezno znanje, kandidat ima postavljen napredni nivo, on dobija osam bodova, što će ga staviti ispred onih koji imaju niži novo tog znanja. Pored obaveznog može biti postavljeno i poželjno znanje koje nosi dodatne bodove kao i obavezno znanje ali ako ga kandidat ne poseduje on će ipak biti prikazan kompanijama, za razliku od slučaja kada ne poseduje obavezno znanje.

Takođe bitno je naglasiti da kompanija mora da izabere oblast rada iz koje joj je potreban kandidat kako bi izvršila pretragu, u suprotnom će joj biti prikazani svi kandidati.

Pošto je objašnjen algoritam bodovanja i rangiranja studenata, sada se može preći na ilustrovanje delova koda koji su izdvojeni kao bitniji.

Sama funkcionalnost ocenjivanja i prikazivanja kandidata kompletno je realizovana kroz klasu **ProcitajStudente**. Prva stvar koja se mora uraditi je čitanje svih studenata i dodeljivanje ocena pomoću kojih će biti rangirani, odnosno izbacivanje nekih studenata sa liste. Ova funkcionalnost je realizovana kroz metodu *FormiranjeListe* koja je prikazana na slici 4.13. Kao što je već navedeno, mora biti podešena oblast rada kandidata, a na slici se vidi da ako kandidat

```
//Formiranje niza koji kasnije uredjujemo zbog citanja parametara
public function FormiranjeListe(){
        $niz=array();
        $konekcija=$this->conn;//pokazivac
        //Prvo citamo sve podatke iz tabele SviStudenti
        $upit=$konekcija->prepare("SELECT * FROM SviStudenti");
        $upit->execute();
        $rezultat=$upit->get_result();
        //Proveravamo da li postoje studenti uopste
        if($rezultat->num_rows>0){
          while($info = $rezultat->fetch_assoc()){
               //Proveravamo  oblast rada
               $obl1=strtolower($this->oblast);
               if($info["$obl1"]==$this->oblast){
                 $upit2=$konekcija->prepare("SELECT *
                                             FROM $obl1
                                             WHERE idstudenta=?");
                 $upit2->bind_param("i",$info["id"]);
                 $upit2->execute();
                 $rezultat2=$upit2->get_result();
                 $info2=$rezultat2->fetch_assoc();
                 //Sada imamo sve informacije
                 $obavezno=$this->ProveriObavezno($info,$info2);
                 //Proveravamo obavezno ako ne zadovoljava izlazimo
                 if($obavezno != "Izadji"){
                   $prosek=$this->ProveriProsek($info);
                   //Proveravamo prosek ako ne zadovoljava izlazimo
                   if($prosek != "Izadji"){
                     $prioritet=$this->ProveriPrioritet($info,$info2);
                     $jezik=$this->ProveriJezik($info);
                     $drugijezik=$this->ProveriDrugiJezik($info);
                     $soft1=$this->ProveriPrviSoftware($info);
                     $soft2=$this->ProveriDrugiSoftware($info);
                     $znanje=$this->ProveriZnanje($info2);
                     $ocena= (int)$obavezno+(int)$prioritet
                            +(int)$jezik+(int)$drugijezik
                            +(int)$soft1+(int)$soft2+(int)$znanje;
                     $id=$info["id"];
                     $pojedinacni=array("id"=>$id,"ocena"=>$ocena);
                     array_push($niz,$pojedinacni);
                     }}}}
```

```
$niz1=$this->NapraviNoviNiz($niz);
if($niz1 == "Nema kandidata."){
                return "Ne";
                                    }
else {
                return $niz1;
        }}}
```

Slika 4.13. Metoda FormirnajeListe

ne zadovoljava oblast rada on odmah ispada iz dalje provere, time se dosta ubrzava pretraga kandidata. Na početku klase je napraviljen jedan niz koji je nazvan *$niz* i u njega će biti smeštani kandidati, tj. njihove ocene i id kandidata koji zadovoljavaju kriterijume pretrage. Kasnije pomoću metode *NapraviNoviNiz* ovaj niz je prvo poslat na uređivanje u metodu *UrediNiz*, a zatim i na filtriranje tako da se dobija niz koji sadrži samo id kandidata u rasporedu po kojem će biti prikazivani kao rezultat pretrage. Ove dve metode su prikazane na slici 4.14. Realizacija ove dve metode je slična pa su zato i prikazane zajedno. Pri izvršavanju metode *NapraviNoviNiz* kao izlaz se dobija niz koji je potreban za čitanje ili obaveštenje da nema kandidata. Ono što je takođe bitno, a što je preskočeno jeste formiranje ocena koje je već opisano u ovom tekstu i sad će biti dat primer kako je to uradjeno u veb aplikaciji (Slika 4.15.). Na ovoj slici je prikazan samo deo koda bodovanja kandidata, a isti princip se nastavlja za sve druge opcije koje mogu biti obavezne, kao i za sve ostale kriterijume ocenjivanja. Kada je izvršena provera svih kriterijuma koje je kompanija unela, a to se radi pozivanjem svih metoda kao što je i prikazano na slici 4.13. (*ProveriPrioritet*, *ProveriJezik*…) formira se konačan broj bodova kandidata i onda se poziva metoda *NapraviNoviNiz* koja je već objašnjena. Kao što je već navedeno, izlazi ovih funkcija su razni nizovi, a konačan niz po kome se vrši čitanje dobija se kao izlaz metode *FormirnanjeListe*. Sada se dolazi do momenta kada je potrebno izvršiti samo čitanje iz baze po nekom određenom redosledu, koji se nalazi u jednom nizu (slika 4.16.). Prva stvar koju je neophodno uraditi je da se niz pretvori u string. Kada je i to urađeno svi potrebni elementi da bi se kandidati čitali i ispisivali su tu i neophodno je samo pozvati određenu funkciju (slika 4.16.).

```
//Metoda za uredjivanje niza
public function UrediNiz($niz){
                        foreach ($niz as $kljuc => $red){
                                $ocena[$kljuc]=$red["ocena"];
                                $id[$kljuc]=$red["id"];
                                                        }

                                array_multisort($ocena,SORT_DESC, $niz);
                                return $niz;
                              }

//Metoda za pravljenje novog niza
public function NapraviNoviNiz($niz){
                              if($niz != NULL){
                                $niz1=$this->UrediNiz($niz);
                                $niz2=array();
                                foreach($niz1 as $kljuc => $red){
```

```
                                $id["kljuc"]=$red["id"];
                                array_push($niz2,$id["kljuc"]);
                                                        }
                        return $niz2;
                                        }
                    else {
                          return "Nema kandidata.";
                          }

                            }
```

Slika 4.14. Uredjivnje i kreiranje novog niza

```
//metoda za proveru obaveznog parametra
public function ProveriObavezno($info,$info2){
       if($this->obavezno != "0"){
         switch($this->obavezno){
               case "jezik":
                 if($this->jezik1 == "0"){ return "Izadji";}
                 else{
                   $jezik=strtolower($this->jezik1);
                   if($info["$jezik"]==$this->jezik1){
                     switch($info["nivo"."$jezik"]){
                           case "Napredni nivo": return 4; break;
                           case "Srednji nivo":  return 3; break;
                           case "Početni nivo":  return 2; break;
                           default:              return 1; break;
                                                    }
                                                      }
                     else{return "Izadji";}}
                       break;
               case "drugijezik":
                 if($this->drugijezik == ""){ return "Izadji";}
                 else{
                   if($info["drugijezik"]==$this->drugijezik) {
                     switch($info["nivodrugijezik"]){
                           case "Napredni nivo": return 4; break;
                           case "Srednji nivo":  return 3; break;
                           case "Početni nivo":  return 2; break;
                           default:              return 1; break;
                                                      }
                     else{return "Izadji";}}
               break;
               case "software1":
                 if($this->soft1 == "0"){ return "Izadji";}
                 else{
                   if($info["$this->soft1"] != NULL
                     and $info["$this->soft1"] != '0'){
                     switch($info["nivo"."$this->soft1"]){
                           case "Napredni nivo": return 4; break;
                           case "Srednji nivo":  return 3; break;
                           case "Početni nivo":  return 2; break;
                           default:              return 1; break;
                                                    } }
```

```
                else{return "Izadji";}}
            break;
```

Slika 4.15. Način bodovanja kandidata

```
public function CitajUslovi(){
            $konekcija=$this->conn;//Pokazivac
            $niz=$this->FormiranjeListe();//Daje nam niz po kom treba da
citamo kandidate iz baze
            if($niz == "Ne" or $niz== NULL){
              echo "<p id='obavestenjepret'>Nemamo odgovarajućeg kandidata
                    u bazi,<br/>
                    molimo Vas ponovite pretragu sa drugim
                    parametrima.</p>";}
            else{
              //Pretvara niz u string zbog citanja iz baze
              $noviniz = implode(", ", $niz);
             //Citanje iz baze po parametrima
             $upit="SELECT imestudenta,id,telekomunikacije,
                      elektronika,energetika,it,automatika,
                    FIELD(id,$noviniz) FROM SviStudenti
                    WHERE id IN ($noviniz)
                    ORDER BY FIELD(id,$noviniz)";
             $rezultat=$konekcija->query($upit);
             if($rezultat->num_rows>0){
               while($row = $rezultat->fetch_assoc()) {
                    $info=(object)$row;
                    echo '<p> <a href="php/student.php?id='.$info->id
                         .'">'.$info->imestudenta .' <span id="obl"> *';
                    $oblasti= array();
                    if($info->telekomunikacije =="Telekomunikacije"){
                      array_push($oblasti,$info->telekomunikacije);
                                                                    }
                     if($info->elektronika=="Elektronika")          {
                         array_push($oblasti,$info->elektronika);
                                                                    }
                     if($info->automatika=="Automatika")            {
                         array_push($oblasti,$info->automatika);
                                                                    }
                     if($info->energetika=="Energetika")            {
                         array_push($oblasti,$info->energetika);
                                                                    }
                     if($info->it=="It")                            {
                         array_push($oblasti,$info->it);
                                                                    }
                     $obl_string=implode(" - ",$oblasti);
                     echo $obl_string;
                     echo '*</span></a></p>';
                                                        }
                                    }
                                    }}
```

Slika 4.16. Čitanje i ispis kandidata po odgovarajućem redosledu

Dodatno se mora naglastiti da je na slici 4.16. uklonjen kompletan deo vezan za “paginaciju”, kako bi mogao biti uredno prikazan deo koji je vezan za ispis kandidata. Kao što se na slici i vidi za čitanje su korišćene funkcije *WHERE id IN* i *ORDER BY FIELD*, prva omogućava da se

iščitavaju samo kandidati čiji se id nalazi u određenom stringu, dok druga obezbedjuje redosled po kom će oni biti pročitani. Dodatni niz nazvan *$oblasti*, koji se vidi na slici 4.16, služi čisto da bi se pedantno ispisale oblasti kojima se student bavi. Kao izlaz celokupne pretrage dobija se odgovarajući kandidati, a grafički izgled ovog dela veb aplikacije je prikazan na slikama 3.7. i 3.8.

Pošto je objašnjen način na koji se dobijaju kandidati pri pretrazi, na redu je deo kada kompanija šalje obaveštenje kandidatu. Ovo je realizovano preko dugmeta "Pošalji obaveštenje" pomoću kojeg se upisuju neke osnovne informacije o kompaniji (id, broj telefona, *e-mail* adresa) i id studenta u tabelu Obavestenja. Kada je sve to upisano u tabelu, pri sledećem prijavljivanju odgovarajućeg studenta na aplikaciju, njemu će na strani obaveštenja biti prikazana informacija o zainteresovanosti kompanije za njega kao potencijalnog kandidata za zapošljavanje ili praksu.

5. ZAKLJUČAK

Pošto su objašnjeni svi glavni delovi veb aplikacije, kako programski tako i sa stanovišta korišćenja, na red je došao rezime cele aplikacije kao i mogućnosti unapređenja iste.

Ova veb aplikacija ima za glavni cilj da kompanijama omogući pronalaženje optimalnog kandidata za zapošljavanje i kao takva je veoma značajna kako za njih tako i za studente i diplomce. Pored toga, ona pruža studentima pomoć pri kreiranju svojih radnih biografija jer ih usmerava ka nekom standardnom tipu istih. Naravno, kod koji je prikazan i objašnjavan u ovom radu je prva verzija veb aplikacije i postoji više mogućnosti unapređenja iste.

Prva stvar koju je moguće unaprediti je naravno algoritam pronalaženja optimalnog kandidata. U ovoj aplikaciji je korišćen identičan algoritam za sve kompanije, a ono što bi moglo biti unapređenje je da se kompanijama u potpunosti omogući kreiranje sopstvenog algoritma za pronalaženje kandidata. Kao alternativa ovom rešenju bi mogla biti studija slučaja koja bi detaljno ispitala sve potrebe pojedinačnih kompanija i onda grupisala rezultate po oblastima kojima se iste bave i na taj način formirala algoritam. Ovo je dosta komplikovana procedura i pitanje je da li bi ona kao takva opravdala rad i uložena sredstva sa stanovišta programera koji piše aplikaciju.

Druga stvar koja bi definitivno bila unaprđivanje ove aplikacije je potiskivanje koda koji je pisan u programskom jeziku *JavaScript* i njegova zamena programskim jezikom *jQuery*. Pored samog izgleda koji je dosta lepši sa stanovišta dizajna, programski jezik *jQuery* bi omogućio i prebacivanje nekih funkcija koje su urađene na serverskoj strani na klijentsku stranu. Ova tranzicija funkcija bi bila od velike pomoći, jer je danas cilj da što više stvari izvršavamo na klijentovom računaru, a što manje na serverima. Opravdanje za tim je potpuno logično jer računar klijenta ima znatno manje operacija koje mora da izvršava dok server ima puno, puno više. Takođe računari klijenata su sve brži tako da je ova tranzicija funkcija trenutno i standard u kreiranju veb aplikacija.

LITERATURA

1. Internet programirnje, Aleksandra Smiljanić - http://home.etf.rs/~aleksandra/IP.html

2. http://www.w3schools.com

3. https://php.net