

Univerzitet u Beogradu

Elektrotehnički fakultet



DIPLOMSKI RAD

REALIZACIJA SOFTVERSKOG SISTEMA ZA EFIKASNO NADGLEDANJE RADA RECENZENATA NAUČNIH PROJEKATA

Mentor

Prof. Dr Boško Nikolić

Student

Isidora Kandić 381/2015

Beograd, 2019.

Sadržaj

1	Uvod	3
2	Zahtevi za realizacijom sistema	5
2.1	Opšti zahtevi	5
2.2	Funkcionalnosti administratora sistema	5
2.3	Funkcionalnosti recenzenta	6
3	Opis korišćenih tehnologija	8
3.1	Java	8
3.2	JSF	8
3.3	Maven	9
3.4	PrimeFaces	10
3.5	Hibernate	10
4	Opis rada sistema	11
4.1	Logovanje i registracija	11
4.2	Korisničko uputstvo – recenzent	14
4.3	Korisničko uputstvo – administator	17
5	Realizacija sistema	22
5.1	Baza podataka	22
5.2	Konfiguracija projekta	24
5.3	Realizacija funkcionalnosti	28
5.4	Problemi prilikom realizacije sistema	33
6	Zaključak	37
7	Literatura	38

1 Uvod

U današnje vreme praktično svaka organizacija, svaki uređen sistem, ima neku vrstu softvera kojim upravlja tim sistemom i svakodnevnim poslovima. U velikom broju slučajeva ti softveri su veb aplikacije jer one često omogućavaju mnogo veću fleksibilnost u radu u odnosu na alternative, uz ništa manje funkcionalnosti.

Softverski sistem za realizaciju baze recenzenata je veb aplikacija koja omogućava funkcionalni pregled i ocenjivanje naučnih radova. Ideja je bila da se na što jednostavniji i intuitivniji način centralizuje proces recenzije naučnih radova. Aplikacija je realizovana iz dva ugla, ugla recenzenta (korisnika) i ugla administratora sistema. Pored osnovnih funkcionalnosti koje se tiču samog ocenjivanja radova, aplikacija omogućava određene dodatne funkcionalnosti za kojima u ovakvim sistemima često postoji potreba.

Prvi tip korisnika, recenzenti, imaju mogućnost pregleda naučnih radova koji su im dodeljeni uz mogućnost ocenjivanja radova kroz odgovore na pitanja povezanog programskog poziva. Pored ovoga, imaju mogućnost praćenja dešavanja u sistemu kroz obaveštenja koja vide na svom profilu i/ili putem elektronske pošte. Da bi pristupilo sistemu, recenzent prvo mora da podnese zahtev za registracijom. Pri tome unosi svoje lične podatke: ime, prezime, nacionalnost, zemlju gde je zaposlen, NIO gde je zaposlen, naučno zvanje, angažovanje, oblast/oblasti za koje je stručan, kontakt podatke (telefon, elektronsku poštu, adresu), ličnu veb stranicu, kao i biografiju (u PDF formatu). Ukoliko mu se zahtev za registracijom odobri, recenzent dobija pristup ostatku sistema. Drugi tip korisnika, administrator, drugi tip korisnika, ima mogućnost pregleda zahteva za registraciju, prihvatanja i odbijanja istih. Administrator upravlja naučnim radovima – može da ih kreira, menja, uništava, dodeljuje određeni projekat recenzentu i oduzima isti. Pored toga, ima mogućnost slanja obaveštenja recenzentima koja mogu biti vidljiva na profilu recenzenta ili putem elektronske pošte.

Kako bi se omogućio ispravan rad aplikacije, pri pokušaju pristupa sistemu od svih korisnika se traži da unesu svoje kredencijale (korisničko ime i lozinku) kako bi se izvršila autentikacija i sprovođenje do odgovarajućeg dela sistema sa kojim dati korisnik ima pravo da interaguje.

Implementacija sistema realizovana je u Java programskom jeziku, koristeći JSF framework. Aplikacija je razdvojena u logičke celine prateći MVC (*Model-View-Controller*) arhitekturu. Gde god je bilo moguće korišćena je Ajax tehnika za optimizaciju učitavanja HTML stranica. Kao pomoć pri realizaciji *front end*-a korišćen je PrimeFaces framework. Pri izradi sistema korišćen je Maven alat, a rađeno je NetBeans razvojnom okruženju.

Prvo poglavlje ovog dokumenta ima za cilj detaljniji opis korisničkih zahteva za realizacijom sistema, kroz nabrojane omogućene opcije za korisnike oba tipa.

U drugom poglavlju opisaće se tehnologije korišćene u svrhu realizacije sistema. U njih spadaju Java programski jezik, JSF radni okvir, PrimeFaces biblioteka, kao druge korišćene biblioteke i alati.

U trećem poglavlju biće opisan rad softverskog sistema u vidu korisničkog uputstva sa odgovarajućim slikama interfejsa i objašnjenim svim implementiranim funkcionalnostima.

Četvrto poglavlje govori o načinu realizacije sistema, počevši od kreiranja adekvatne baze podataka, konfiguracije projekta, pa do načina implementacije svih potrebnih zahteva. Gde je smatrano za shodno biće navedeni delovi koda koji rešavaju karakteristične probleme. Takođe, biće izdvojeni problemi na koje se nailazilo tokom razvoja aplikacije, kao i rešenja odabrana za prevazilaženje istih.

Kao zaključak, pored rekapitulacije samog rada, diskutovaće se o eventualnim poboljšanjima sistema i navesti moguća unapređenja.

2 Zahtevi za realizacijom sistema

Ovo poglavlje definiše korisničke zahteve sistema sa detaljima svih omogućenih funkcionalnosti.

2.1 Opšti zahtevi

Aplikacija je predviđena za korišćenje od strane dva tipa korisnika, recenzenata i administratora sistema. Svaki od korisnika se najpre uloguje korišćenjem svojih kredencijala putem forme na početnoj stranici da bi, u zavisnosti od uloge koja mu je dodeljena, nastavio rad sa sistemom sa odgovarajućim pravima pristupa.

Ukoliko recenzent nije prethodno registrovan u sistem nudi se mogućnost registracije novog korisnika. Smatra se da su administratorski nalozi predefinisani u samoj bazi podataka. Nakon uspešnog logovanja, svim korisnicima je omogućeno odjavljivanje sa sistema u bilo kom trenutku rada. Recenzenti imaju mogućnost pregleda svog profila i menjanja ličnih podataka.

U slučaju bilo kog tipa greške, kao što su nevalidnost unetih podataka ili nemogućnost izvršavanja tražene radnje iz bilo kog razloga, korisnik biva adekvatno obavešten prikazivanjem adekvatne poruke o grešci na tekućoj stranici.

2.2 Funkcionalnosti administratora sistema

Uloga administratora je da upravlja procesom kreiranja, dodele i ocenjivanja naučnih radova, zatim da upravlja zahtevima za registraciju recenzenata, kao i da vrši obaveštavanje recenzenata. Od funkcionalnosti ima sledeće:

- Logovanje – unošenjem predefinisano korisničkog imena i lozinke. U slučaju pogrešno unetih podataka ispisuje se odgovarajuća poruka o grešci.
- Odjavljivanje – korisnik završava rad sa sistemom i preusmerava se na početnu stranicu radi ponovnog logovanja ili registracije novog člana.
- Pregled zahteva za registraciju – u sekciji Zahtevi za registraciju ima pregled neobraćenih zahteva sa izlistanim svim podacima potencijalnog recenzenta, uz mogućnost prihvatanja ili odbijanja istog. Pri prihvatanju/odbijanju zahteva, recenzentu se na adresu elektronske pošte šalje obaveštenje o ishodu.

- Kreiranje naučnog projekta – administrator jedini ima mogućnost definisanja naučnog projekta. Pri tome se definiše: naziv projekta, rukovodilac projekta, NIO rukovodioca projekta, njegovo zvanje i angažovanje, prijava projekta koja se sastoji od nekoliko dokumenata (opis projekta, opis aktivnosti na projektu, biografije članova tima, budžet, itd.), oblast kojoj projekat pripada, kao i datum podnošenja prijave projekta.
- Izmena i brisanje naučnog projekta – administrator jedini ima mogućnost modifikovanja bilo koje informacije o projektu
- Dodela naučnog projekta recenzentu – administrator je zadužen za povezivanje naučnog projekta i recenzenta, kao i za eventualno oduzimanje projekta recenzentu. Pri dodeli se beleži datum dodeljivanja
- Upravljanje programskim pozivima – administrator ima mogućnost unosa, promene i brisanja programskih poziva, kao i pitanja koja su povezana sa ocenjivanjem poziva. Pozivi se kasnije povezuju sa odgovarajućim naučnim radom.
- Praćenje rada recenzenata – administratoru je omogućen pregled broja neocenjenih radova svakog recenzenta, kao i upoređivanje ocene recenzenta sa konačnom odlukom za dati projekat
- Slanje obaveštenja – administrator može slati obaveštenja recenzetima na dva načina: tako da ona budu vidljiva na profilu recenzenta i slanjem obaveštenja na adresu elektronske pošte recenzenta

2.3 Funkcionalnosti recenzenta

Uloga recenzenta je pregled i ocenjivanje naučnih radova koji su mu dodeljeni odgovaranjem na set pitanja koja pripadaju programskom pozivu kome naučni projekat koji se ocenjuje pripada. Funkcionalnosti koje ovaj korisnik ima su sledeće:

- Registracija – da bi mogao da interaguje sa sistemom recenzent se mora prvo registrovati. Pri tome unosi svoje lične podatke: ime, prezime, nacionalnost, zemlju gde je zaposlen, NIO gde je zaposlen, naučno zvanje, angažovanje, oblast/oblasti za koje je stručan, kontakt podatke (telefon, elektronsku poštu, adresu), ličnu veb stranicu, kao i biografiju.

- Praćenje stanja zahteva za registraciju – pre nego što administrator obradi njegov zahtev korisnik može na svom profilu da prati stanje prijave (stigla, razmatra se, prihvaćena, odbijena)
- Logovanje – vrši se unošenjem korisničkog imena i lozinke odabranih prilikom registracije. U slučaju pogrešno unetih podataka ispisuje se odgovarajuća poruka o grešci. U slučaju zaboravljene lozinke na registrovanu adresu elektronske pošte se šalje privremena lozinka koju korisnik može iskoristiti da resetuje lozinku.
- Odjavljivanje – korisnik završava rad sa sistemom i preusmerava se na početnu stranicu radi ponovnog logovanja ili registracije novog člana.
- Pregled profila – ima mogućnost pregleda ličnih podataka unetih prilikom registracije.
- Izmena profila – mogućnost modifikacije ličnih podataka.
- Pregled svih svojih projekata – recenzent ima mogućnost pregleda i prispupa podacima svih naučnih projekata koji su mu dodeljeni
- Ocenjivanje naučnih projekata – sve neocenjene projekte recenzent može oceniti, pri čemu mu se traži da odgovori na set pitanja odgovarajućeg programskog poziva. Moguće je sačuvati i zaključati odgovore, što potrazumeva slanje odgovora na dalju obradu i nemogućnost promene istih. Pored toga, moguće je i samo sačuvati odgovore u sistemu, koje recenzent može kasnije promeniti pre slanja na dalju obradu.
- Pregled obaveštenja – kada se uloguje u sistem korisniku se prikazuje stranica sa najnovijim obaveštenjima koje je administrator poslao.

3 Opis korišćenih tehnologija

Ovo poglavlje nudi prikaz tehnologija korišćenih za razvoj aplikacije, uz objašnjenja zašto se baš za njih odlučilo.

Kao razvojno okruženje za pisanje Java koda korišćen je NetBeans IDE 8.2 koje je vrlo pogodno za kreiranje veb aplikacija iz razloga što nudi brojna olakšanja pri razvoju aplikacije, kao što je automatsko generisanje konfiguracionih fajlova. Uz to, nudi razvoj aplikacije uz korišćenje integrisanog Maven alata za jednostavno importovanje eksternih biblioteka. O svemu tome će biti više reči u nastavku. Kao radni okvir za veb aspekt softverskog sistema korišćen je JSF (*Java Servlet Faces*) framework.

Kao sistem za upravljanje bazom podataka korišćen je MySQL, sa svojim serverom i MySQL Workbench 8.0 radi lakšeg testiranja. Za konekciju aplikacije sa tako kreiranom bazom korišćen MySQL JDBC (*Java Database Connectivity*) i Hibernate kao implementacija JPA (*Java Persistence API*).

Za izradu fleksibilnih veb-stranica korišćena je PrimeFaces biblioteka, a radi optimizacije korišćena je Ajax tehnologija gde god je to bilo moguće. Google Chrome brauzer je odabran kao veb pretraživač u svrhe testiranja aplikacije, ali softver je kompatibilan sa svim modernim veb pretraživačima.

3.1 Java

Java je opšte namenski, objektno orijentisani, konkurentni, klasno orijentisani programski jezik koji je platformski nezavistan. Upravo je platformska nezavisnost najveća prednost Java programskog jezika i razlog zašto je jedan od najpopularnijih jezika širom sveta. Platformska nezavisnost se ostvaruje tako što se izvorni kod ne kompajlira u mašinski kod direktno, već u takozvani bajt kod. Bajt kod je mašinski kod Javine virtuelne mašine (JVM) koja u vreme izvršavanja taj kod prevodi u izvršni kod mašine na kojoj se izvršava. Direktna posledica ovoga je princip “piši jednom izvršavaj svuda” (WORA – *Write Once, Run Anywhere*), koji je bio ključan za ekspanziju Interneta zato što je omogućio da se isti Java kod izvršava na bilo kojoj mašini širom sveta. Deo Java koda može da se izvršava i u samom veb brauzeru pomoću Java Applet-a koji predstavlja mini Java program dinamički učitani u pretraživač. (Schildt, 2014).

3.2 JSF

JSF (*JavaServer Faces*) je radni okvir za razvoj Java veb aplikacija koji ima za cilj pojednostavljenje procesa razvoja i integracije korisničkog interfejsa sa serverskom stranom aplikacije. JSF definiše standardizovan način izgradnje prikaza veb stranica za postojeće serverske

aplikacije pomoću reupotrebljivih ulazno-izlaznih komponenti koje znato pojednostavljaju sve operacije zahtevane za taj poduhvat. (Oracle, 2019)

JSF smanjuje olakšava kreiranje i održavanje aplikacija koje će se izvoditi na Java aplikacionom serveru i koje će usmeravati korisničko sučelje aplikacije ciljnom klijentu. JSF olakšava razvoj veb aplikacija tako što:

- Obezbeđuje korisničke UI (ulazno-izlazne) komponente koje su pogodne za višekratnu upotrebu
- Omogućava jednostavan prenos podataka između UI komponenti
- Omogućavanje implementacije kastimizovanih komponenti
- Povezivanje događaja na strani klijenta sa kodom aplikacije na strani servera

Sam razvojni okvir se sastoji iz: osnovne biblioteke, skupa osnovnih UI komponenti koji predstavljaju standardne HTML komponente, skupa UI komponenti koje predstavljaju ekstenziju osnovnih i koje omogućavaju nove funkcionalnosti, kao i skupa funkcionalnosti koje omogućavaju uslovno generisanje UI komponenti na veb stranici. Ovo poslednje je značajno zato što omogućava da ista stranica ima različit izgled i funkcionalnosti u zavisnosti od tipa korisnika, stanja sistema ili nekog drugog faktora, što softverski sistem čini znatno dinamičnijim i fleksibilnijim.

Povezivanje JSF veb stranice sa serverskim kodom omogućen je korišćenjem takozvanih ManagedBean-ova koji su ništa drugo nego obični JavaBean-ovi registrovani u JSF-u. ManagedBean predstavlja kontrolersku klasu koja sadrži privatna polja sa javnim getter i setter metodama koja se mogu direktno povezati sa JSF veb stranicom, kao i javne metode koje se takođe pozivaju direktno iz koda JSF veb stranice.

JSF framework je baziran na MVC arhitekturi koja se koristi za razdvajanje logike sistema od prezentacije. Modularnost sistema i razdvajanje logičkih celina u što nezavisnije podsisteme je ključna za razvoj svih kompleksnih sistema kako bi različiti programeri mogli paralelno da razvijaju različite delove sistema (TutorialsPoint, 2019).

3.3 Maven

Maven je alat koji služi za upravljanje softverkim sistemom, kao i olakšani razvoj i razumevanje istog. Konfiguracija alata radi se na osnovu POM (*Project Object Model*) fajla u kome se definišu neke osnovne osobine projekta kao što su naziv i verzija jar, odnosno war, fajla u koji se dati kod transformiše. Još značajnije od toga, u pom.xml fajlu deklarišu se i sve zavisnosti potrebne za pokretanje aplikacije. Na ovaj način sve neophodne biblioteke programer nije u obavezi ručno da uključi u projekat, već to za njega radi Maven. To je omogućeno pomoću lokalnog repozitorijuma, tačnije skladišta .jar fajlova, koje Maven pretražuje prilikom svakog pokretanja u potrazi za zahtevanim artifaktima u pom.xml fajlu, dok u slučaju da ih ne nađe, iste preuzima sa centralnog repozitorijuma sa interneta i smešta ih u lokalni (The Apache Software

Foundation, 2002). Ovim je kompletno upravljanje eksternim zavisnostima prebačeno u odgovornost alata, što je i glavni razlog opredeljenja za kreiranje projekta pomoću Maven alata.

3.4 PrimeFaces

PrimeFaces je open source radni okvir za razvoj JSF aplikacija. Sastoji se od preko sto UI komponenti koje predstavljaju nadogradnju na standardne JSF komponente. PrimeFaces je dakle, još jedan sloj iznad JSF-a koji podržava različite teme za grafički prikaz koje čine da veb stranice izgledaju sređenije i elegantnije. Još značajnije, PrimeFaces komponente implemntiraju Ajax tehnologiju za osvežavanje prikaza, maksimalno optimizujući performanse dela aplikacije zadužene za prikaz. Pored poboljšanih standardnih JSF UI komponenti, PrimeFaces nudi i dodatne UI komponente kao integrisano rešenje za često tražene prikaze, za koje bi inače bilo potrebno koristiti više različitih JSF UI komponenti.

3.5 Hibernate

Hibernate je radni okvir koji predstavlja implementaciju JPA (*Java Persistence API*) i služi za omogućavanje komunikacije sa bazom podataka. Pored mogućnosti povezivanja sa bazom i izvršavanja upita nad istom, Hibernate omogućava ORM (*Object/Relational Mapping*) – mapiranje Java objekata u klasne entite, kao i obrnutno. U svrhu automatizacije najosnovnijih CRUD (Create, Read, Update, Delete) operacija Hibernate se služi HQL (Hibernate Query Language) upitnim jezikom, koji je na prvi pogled jako sličan SQL-u. Međutim, za razliku od njega, HQL je u potpunosti objektno orijentisan i razume pojmove poput nasleđivanja i polimorfizma (Bauer, King & Gregory, 2015).

4 Opis rada sistema

Ovo poglavlje nudi korisničko uputstvo za oba tipa korisnika, sa priloženim slikama interfejsa i detaljnim opisom implementiranih funkcionalnosti.

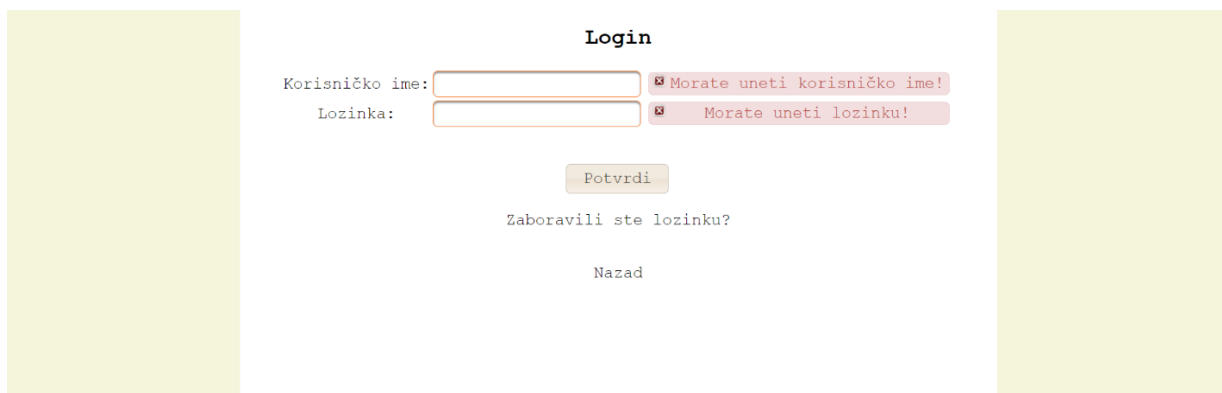
4.1 Logovanje i registracija

Početna stranica aplikacije sadrži linkove ka stranici za registraciju, odnosno logovanje (Slika 4.1.1).



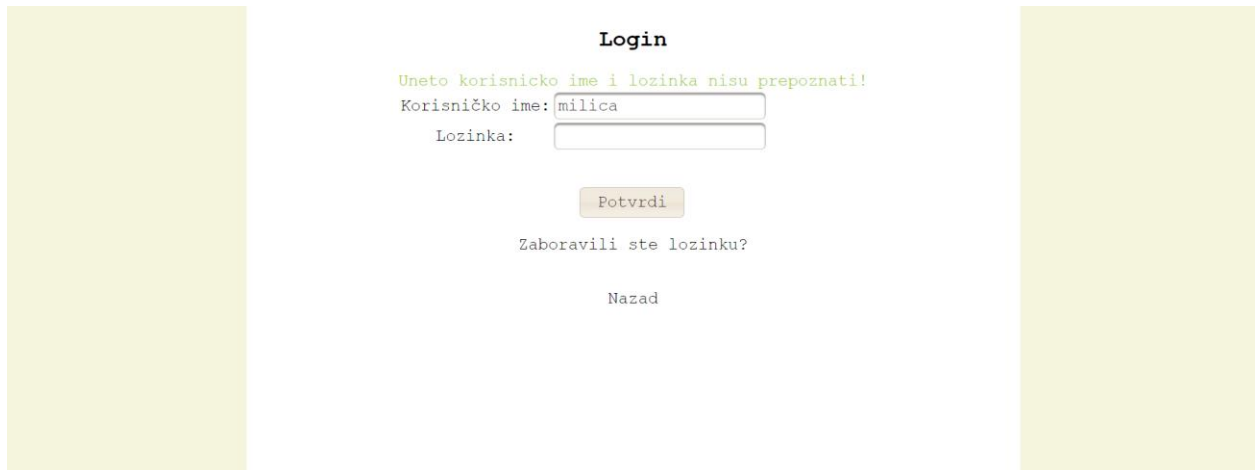
Slika 4.1.1

Klikom na link Login korisnik biva preusmeren na stranicu za logovanje u sistem. Od njega se traži da unese svoje kredencijale u vidu korisničkog imena i lozinke. Oba polja su obavezna i u slučaju da se ne unese vrednost ispisuje se poruka o grešci (Slika 4.1.2).



Slika 4.1.2

U slučaju da kombinacija korisničkog imena i lozinke nije prepoznata u sistemu ispisuje se odgovarajuća poruka korisniku (Slika 4.1.3).



Login

Uneto korisnicko ime i lozinka nisu prepoznati!

Korisničko ime:

Lozinka:

[Zaboravili ste lozinku?](#)

[Nazad](#)

Slika 4.1.3

U slučaju da je korisnik zaboravio lozinku nudi mu se mogućnost da traži od sistema da mu privremenu lozinku pošalje na adresu elektronske pošte (Slika 4.1.4), nakon čega mu se automatski generisana privremena lozinka šalje na unetu adresu (Slika 4.1.5). Uslov je naravno da je ta adresa identična adresi korisnika koja je zabeležena u sistemu.



Promena lozinke

Molim Vas unesite svoje korisnicko ime i e-mail adresu koju ste koristili pri registraciji kako bismo Vam poslali privremenu lozinku.

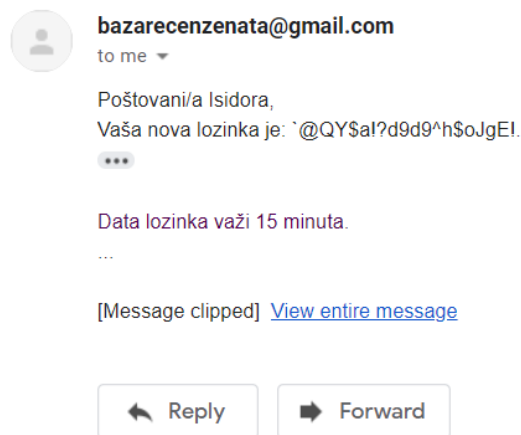
Korisnicko ime:

E-mail:

[Promeni lozinku](#)

[Nazad](#)

Slika 4.1.4



Slika 4.1.5

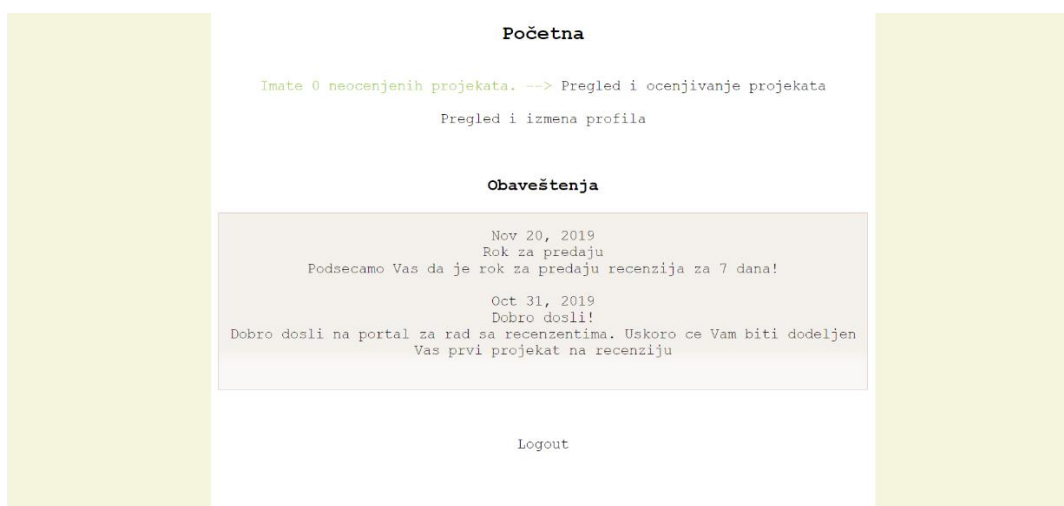
Novi korisnici mogu poslati zahtev za registracijom pri čemu unose tražene podatke (Slika 4.1.6). Od tih su samo ime, prezime, korisničko ime, lozinka, potvrda lozinke i adresa elektronske pošte obavezni.

A screenshot of a web registration form titled 'Registracija'. The form contains the following fields: 'Ime:' (Isidora), 'Prezime:' (Kandić), 'Korisničko ime:' (isidora), 'Lozinka:' (masked with dots), 'Potvrda lozinke:' (masked with dots), 'Nacionalnost:' (srpska), 'Zemlja zaposlenja:' (Srbija), 'NIO zaposlenja:' (empty), 'Naučno zvanje:' (naučni savetnik), 'Angažovanje:' (empty), 'Stručne oblasti: (razdvajati zarezom)' (java, jsf), 'Mobilni telefon:' (empty), 'E-mail:' (isidora.kandic@gmail), 'Adresa:' (empty), 'Lična veb stranica:' (empty), and 'Biografija:' (with a file upload button 'Izaberi fajl' and a file name 'CV.pdf'). At the bottom are two buttons: 'Potvrdi' and 'Nazad'.

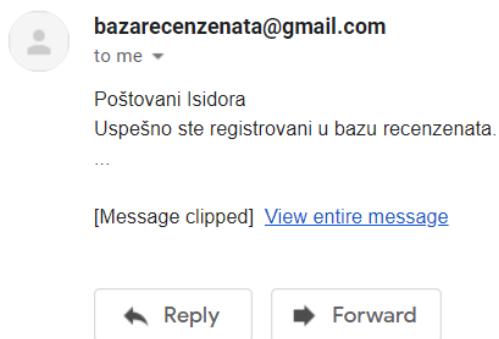
Slika 4.1.6

4.2 Korisničko uputstvo – recenzent

U slučaju uspešne registracije korisnici čije su prijave za registraciju odobrene bivaju preusmereni na početnu stranu za rad recenzenata gde im se prikazuju obaveštenja, broj neocenjenih projekata, a imaju i mogućnost biranja linka ka funkcionalnosti koja ih zanima (Slika 4.2.1). Onim korisnicima čiji zahtevi još uvek nisu odobreni ispisuje poruka o statusu njihovog zahteva za registracijom – da li još uvek čeka obradu, ili je odbijen. Kada administrator odbije ili prihvati zahtev za registraciju korisnik putem mejla biva obavešten o tome (Slika 4.2.2).



Slika 4.2.1



Slika 4.2.2

Klikom na link Pregled i ocenjivanje projekata recenzentu se prikazuje tabela sa svim njemu dodeljenim projektima koja sadrži pregled detalja projekta (Slika 4.2.3). U polju Ocena nalazi se

link sa pitanjima programskog poziva datog projekta, kao i polje za unos konačne ocene za projekat (Slika 4.2.4). U slučaju da je korisnik već odgovorio na pitanja, ali ih nije poslao već samo zapamtio, ti odgovori su mu vidljivi kad se ovaj prozor otvori. Na dalje mu se pruža mogućnost da ih menja i ponovo zapamti ili zapamti i pošalje. Poslate odgovore nije moguće menjati tako da se za projekte čiji su odgovori poslani klikom na Ocena otvara ista veb stranica kao za ocenjivanje ali je izmena odgovora onemogućena. Klikom na Pregled dokumenata na istoj stranici se ispod tabele prikazuju nazivi i sadržaji svih dokumentata projekta.

Pregled projekata								
Naziv	Rukovodilac	NIO rukovodioca	Zvanje rukovodioca	Angazovanje rukovodioca	Oblast	Dokumenta	Stanje	
Baza podataka interneta	Petar Petrovic	ETF	dipl. inz.	profesor	programiranje	Pregled dokumenata	nema	Ocena
Softverski sistem za resenje svih problema	Marko Markovic	ETF	dipl. inz.	profesor	programiranje sve	Pregled dokumenata	nema	Ocena

Nazad

Slika 4.2.3

Ocenjivanje projekata

Koliko je projekat isplativ?

☐ malo
☒ vrlo
☐ srednje

Da li je projekat inovativan?

☐ ne
☒ da

Ocena projekta:

Nazad

Slika 4.2.4

Link Pregled i izmena profila vodi korisnika na stranicu na kojoj su mu prikazani njegovi lični podaci u formi sličnoj formi za registraciju tako da korisnik može promeniti vrednost polja koja želi i klikom na dugme Izmeni profil zapamtiti te promene (Slika 4.2.5). Ako se želi izvršiti promena lozinke to se radi klikom na dugme Promeni lozinku koji korisnika preusmerava na stranicu na kojoj može to učiniti (Slika 4.2.6).

Vaš profil

Ime:

Prezime:

Korisničko ime:

Nacionalnost:

Zemlja zaposlenja:

NIO zaposlenja:

Naučno zvanje:

Angažovanje:

Stručne oblasti: (razdvajati zarezom)

Mobilni telefon:

E-mail:

Adresa:

Lična veb stranica:

Biografija:

Slika 4.2.5

Promena lozinke

Trenutna lozinka:

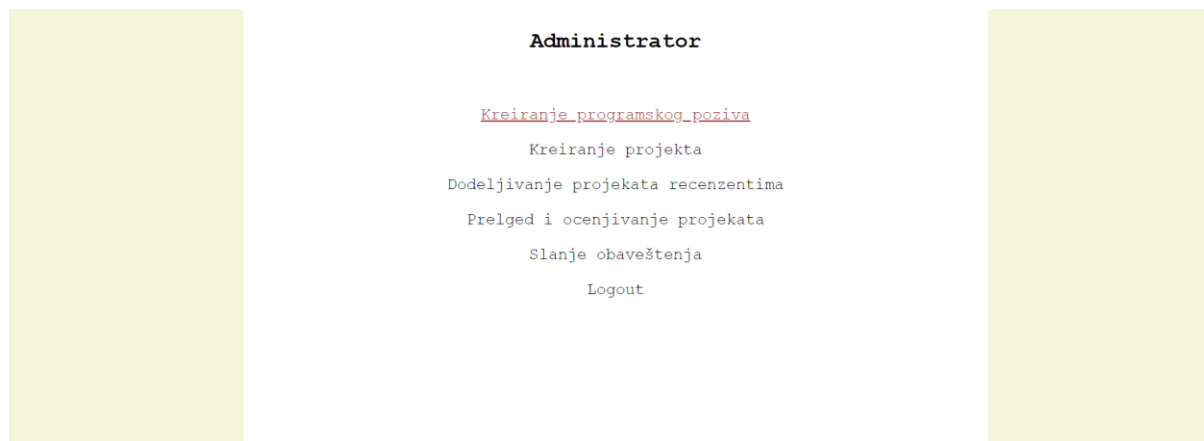
Nova lozinka:

Potvrda lozinke:

Slika 4.2.6

4.3 Korisničko upustvo – administrator

Administratoru se na početnoj strani prikazuje lista linkova ka funkcionalnostima koje su mu omogućene (Slika 4.3.1). Klikom na link kreiranje programskog poziva administrator biva preusmeren na stranicu na kojoj može da unese prvo naziv programskog poziva, zatim da dodaje jedno po jedno pitanje istog i jedan po jedan odgovor (Slika 4.3.2). Klikom na dugme Dalje prelazi se na unos sledećeg pitanja odnosno odgovora. Rad se završava klikom na dugme Završi unos.



Slika 4.3.1



Slika 4.3.2

Kreiranje projekta vrši se u dva koraka. U prvom se korisniku otvara forma za unos potrebnih osnovnih informacija o projektu, kao i referenciranje programskog poziva kom projekat pripada (Slika 4.3.3), a u drugom delu korisnik ima mogućnost učitavanja proizvoljnog broja dokumenata projekta koja će biti vidljiva recenzentima pri ocenjivanju (). Kreirane projekte administrator može dodeljivati (i oduzimati) recenzentima kroz posebnu formu u kojoj sparuje parove recenzent-projekat (Slika 4.3.4).

Kreiranje naučnog projekta

Naziv:

Rukovodilac:

NIO rukovodioca:

Zvanje rukovodioca:

Angazovanje rukovodioca:

Oblast:

Programski poziv:

Slika 4.3.3

Kreiranje naučnog projekta

Dokumenta:

Module 1_Preparing for an exam certificate.pdf	219.3 KB	<input type="text"/>	<input type="button" value="✖"/>
Module 2_Running an exam certificate.pdf	219.3 KB	<input type="text"/>	<input type="button" value="✖"/>
Module 3_Dealing with malpractice, incidents and inspections certificate.pdf	219.5 KB	<input type="text"/>	<input type="button" value="✖"/>

Slika 4.3.4

Pregled dodeljenih projekata korisniku prikazuje sve dodeljene projekte u vidu naziva recenzenta, naziva projekta, datuma dodeljivanja, ocenu recenzenta, kao i trenutnu konačnu ocenu za projekat dodeljenu od strane administratora. Na istoj stranici pružena je mogućnost dodeljivanja te konačne ocene (Slika 4.3.5). Sama ocena recenzenta je zapravo link koji kada se klikne u novom prozoru otvara pregled odgovora recenzenta na pitanja programskog poziva (Slika 4.3.6), koji administratoru treba da pomognu pri donošenju konačne odluke o projektu.

Pregled dodeljenih projekata

Recenzent	Projekat	Datum dodeljivanja	Ocena recenzenta	Konačna ocena
isidora	Softverski sistem za resenje svih problema	2019-12-10	56	60
isidora	Baza podataka interneta	2019-12-10	77	

Donošenje konačne odluke o projektu

Naziv projekta:

Konačna ocena:

Slika 4.3.5

Pregled ocena

Recenzent: isidora
Projekat: Softverski sistem za resenje svih problema

Koliko je projekat isplativ?

☒ malo
☐ srednje
☐ vrlo

Da li je projekat inovativan?

☐ ne
☒ da

Ocena projekta:

Slika 4.3.6

Jedna od funkcionalnosti administratora je mogućnost slanja obaveštenja recenzentima. Administrator unosi naslov, tekst i potpis obaveštenja, zatim bira da li želi da pošalje obaveštenje na mejling listu, ili da ono bude vidljivo na samom profilu svakog od recenzenata (Slika 4.3.7).

Administrator ima pregled svih korisnika u sistemu: registrovanih, neregistrovanih i onih čiji zahtevi za registraciju nisu još uvek obrađeni i to na objedinjenom prikazu. Registrovani korisnici su obojeni zelenom bojom i njih se prikazuje broj neocenjenih projekata koje trenutno imaju na recenziji, odbijeni zahtevi su obojeni zelenom bojom, dok se za neobrađene zahteve nudi mogućnost prihvatanja ili odbijanja zahteva za registraciju (Slika 4.3.8).

Kreiranje obaveštenja

Naslov:

Teskt:

Svim recenzentima želimo srećnu Novu godinu i Božićne praznike i nadamo se da ćemo još bolje i plodonosnije saradivati u godini pred nama!

Potpis:

Tip obaveštenja: ☒ mejl ☐ profil

Slika 4.3.7

Pregled svih korisnika

Ime	Prezime	Korisnicko ime	Broj neocenjenih projekata		
Marko	Markovic	marko	-	<input type="button" value="Prihvati"/>	<input type="button" value="Odbij"/>
Sofija	Kandic	sofija	-	<input type="button" value="Prihvati"/>	<input type="button" value="Odbij"/>
Isidora	Kandic	isidora	0	<input type="button" value="Prihvati"/>	<input type="button" value="Odbij"/>

[Nazad](#)

Slika 4.3.8

Pored pregleda korisnika administrator ima mogućnost pregleda detalja svih projekata, kao i mogućnost izmena i brisanja projekta (Slika 4.3.9). Klikom na link pregled dokumentata korisniku se u istom prozoru ispod tabele prikazuju sva dokumenta projekta. Klikom na dugme izmeni korisnik biva preusmeren na novu stranicu sa formom sličnom formi za kreiranje novog projekta gde može modifikovati željene podatke, kao i pridružiti projektu nova dokumenta. Opcija obriši briše projekat kako iz date liste, tako iz samog sistema.

Pregled projekata								
Naziv	Rukovodilac	NIO rukovodioca	Zvanje rukovodioca	Angazovanje rukovodioca	Oblast	Dokumenta		
Moj projekat	Isidora					Pregled dokumenata	Izmeni	Obrisi
Moj projekat 2	neko		još ništa			Pregled dokumenata	Izmeni	Obrisi
Softverski sistem za rešenje svih problema	Marko Markovic	ETF	dipl. inz.	profesor	programiranje sve	Pregled dokumenata	Izmeni	Obrisi
Baza podataka interneta	Petar Petrovic	ETF	dipl. inz.	profesor	programiranje	Pregled dokumenata	Izmeni	Obrisi

Slika 4.3.9

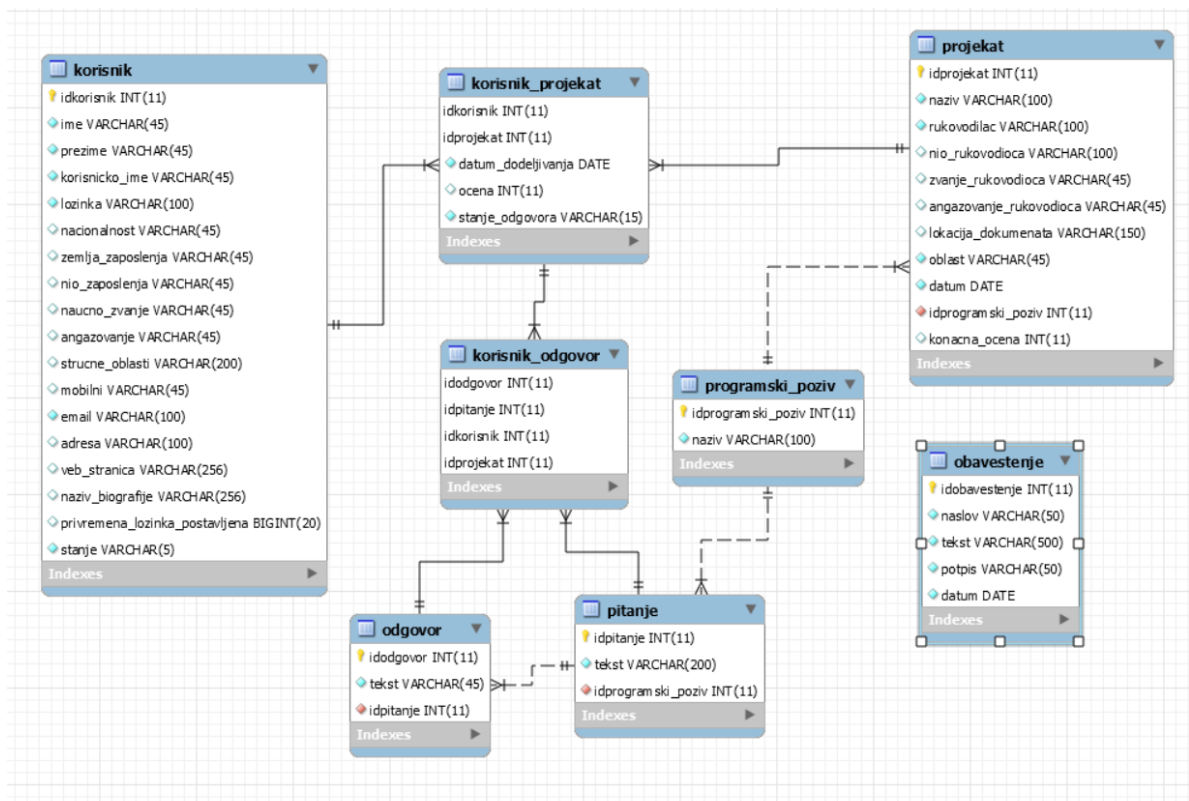
5 Realizacija sistema

U ovom poglavlju biće objašnjen način kreiranja i realizacije veb aplikacije počevši od neophodne baze podataka za rad sistema, preko konfiguracije samog projekta, do načina implementiranja zahtevanih funkcionalnosti. Sve to biće ilustrovano primerima samog koda. Biće navedeni i problemi na koje se naišlo tokom rada i rešenje za koje se odlučeno da se primene.

5.1 Baza podataka

Za potrebe administriranja MySQL sistema baze podataka korišćen je MySQL 5.7 server sa predefinisanim korisničkim imenom *root* i lozinkom *root*. Za kreiranje baze, popunjavanje baze, kao i pregled sadržaja baze tokom razvoja i testiranja aplikacije korišćen je MySQL Workbench 8.0 kao alat koji kroz grafički korisnički interfejs znatno olakšava komunikaciju sa bazom.

Relaciona šema baze podataka softverskog rešenja prikazana je u nastavku (Slika 5.1.1).



Slika 5.1.1

Portabilnost baze podataka MySQL Workbench omogućava tako što nudi mogućnost kreiranja takozvanog dump-a baze u kome se može nalaziti samo SQL kod za kreiranje baze, ali i kod za kreiranje i popunjavanje baze. Primer automatski generisanog SQL koda za kreiranje i popunjavanje tabele korisnik dat je u nastavku (Slika 5.1.2).

```

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `korisnik`
--

DROP TABLE IF EXISTS `korisnik`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `korisnik` (
  `idkorisnik` int(11) NOT NULL AUTO_INCREMENT,
  `ime` varchar(45) CHARACTER SET latin7 NOT NULL,
  `prezime` varchar(45) CHARACTER SET latin7 NOT NULL,
  `korisnicko_ime` varchar(45) CHARACTER SET latin7 NOT NULL,
  `lozinka` varchar(100) CHARACTER SET latin7 NOT NULL,
  `nacionalnost` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `zemlja_zaposlenja` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `nio_zaposlenja` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `naučno_zvanje` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `angazovanje` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `strucne_oblasti` varchar(200) CHARACTER SET latin7 DEFAULT NULL,
  `mobilni` varchar(45) CHARACTER SET latin7 DEFAULT NULL,
  `email` varchar(100) CHARACTER SET latin7 NOT NULL,
  `adresa` varchar(100) CHARACTER SET latin7 DEFAULT NULL,
  `veb_stranica` varchar(256) CHARACTER SET latin7 DEFAULT NULL,
  `naziv_biografije` varchar(256) CHARACTER SET latin7 DEFAULT NULL,
  `privremena_lozinka_postavljena` bigint(20) DEFAULT '0',
  `stanje` varchar(5) CHARACTER SET latin7 NOT NULL,
  PRIMARY KEY (`idkorisnik`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `korisnik`
--

```

Slika 5.1.2

Konekcija sa bazom konfigurisana je u *hibernate.cfg.xml* fajlu. Bilo je potrebno navesti korišćeni drajver za bazu, tačan URL konekcije sa imenom same baze podataka, korisničko ime i lozinku, kao i naglasiti da baza funkcioniše na MySQL dijalektu (Slika 5.1.3).

```

<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/baza_recenzenata?useSSL=false</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>
  </session-factory>
</hibernate-configuration>

```

Slika 5.1.3

5.2 Konfiguracija projekta

Osnovne konfiguracije uradilo je samo razvojno okruženje, *NetBeans IDE 8.2* kada se pri kreiranju projekta reklo da se želi kreirati veb aplikacija koja koristiti JSF radni okvir i *GlassFish 4.1.1* kao veb server za potrebe testiranja. Tom prilikom automatski su generisani fajlovi za konfiguraciju servera, *glassfish-web.xml* (Slika 5.2.1) i *glassfish-resources.xml* (Slika 5.2.2), kao i konfiguracioni fajl koji svaka Java veb aplikacija mora da poseduje kako bi bilo omogućeno postavljanje iste na server – *web.xml*. Konfiguracioni fajl *web.xml* modifikovan je u toku daljeg rada kada se odlučilo da se za prikaz veb stranica koristi PrimeFaces radni okvir i tada su dodate potrebne konfiguracije i za to (Slika 5.2.3).

U aplikaciji je urađen ORM čime su entiteti iz baze mapirani u Java klase. Ovaj deo je takođe automatski urađen korišćenjem *NetBeans IDE 8.2* vizarda za *Reverse Ingeneering* i *Mapping POJOs from Database*.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org/DTD GlassFish Application Server 3.1 Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app 3 0-1.dtd">
<glassfish-web-app>
  <class-loader delegate="false"/>
</glassfish-web-app>

```

Slika 5.2.1


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN" "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attempts="0"
    connection-creation-retry-interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-seconds="0"
    connection-validation-method="auto-commit" datasource-classname="com.mysql.jdbc.jdbc2.optional.MysqlDataSource"
    fail-all-connections="false" idle-timeout-in-seconds="300" is-connection-validation-required="false"
    is-isolation-level-guaranteed="true" lazy-connection-association="false" lazy-connection-enlistment="false"
    match-connections="false" max-connection-usage-count="0" max-pool-size="32" max-wait-time-in-millis="60000"
    name="mysql_baza_recenzenata_rootPool" non-transactional-connections="false" pool-resize-quantity="2"
    res-type="javax.sql.DataSource" statement-timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-seconds="0"
    wrap-jdbc-objects="false">
    <property name="serverName" value="localhost"/>
    <property name="portNumber" value="3306"/>
    <property name="databaseName" value="baza_recenzenata"/>
    <property name="User" value="root"/>
    <property name="Password" value="root"/>
    <property name="URL" value="jdbc:mysql://localhost:3306/baza_recenzenata?zeroDateTimeBehavior=convertToNull"/>
    <property name="driverClass" value="com.mysql.jdbc.Driver"/>
  </jdbc-connection-pool>
  <jdbc-resource enabled="true" jndi-name="java:app/baza_recenzenata" object-type="user" pool-name="mysql_baza_recenzenata_rootPool"/>
</resources>
```

Slika 5.2.2

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>humanity</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

Slika 5.2.3

Sve konfiguracije koje se tiču eksternih biblioteka navedene su *pom.xml* fajlu iz kog Maven alat čita u uvezuje potrebne fajlove. Korišćene zavisnosti su:

- mysql-connector-java – konektor potreban za povezivanje sa MySQL bazom
- hibernate-agroal – neophodne biblioteke za funkcionisanje Hibernate-a

- hibernate-entitymanager – iako se u korisničkom kodu povezivanje sa bazom vršilo korišćenjem *SessionFactory* objekta, a ne preko *EntiyManager*-a, vizard korišćen za ORM zahteva postojanje *EntiyManager*-a kako bi ispravno funkcionisao
- javaee-web-api – potrebno za realizaciju brojnih funkcionalnosti vezanih za ManagedBean-ove i funkcionisanje veb aplikacije uopšte
- hibernate-java8 – pomoćna zavisnost koja omogućava mapiranje SQL datuma u java.util datum i obrnuto
- commons-codec – korišćeno za enkripciju lozinke
- primefaces – zavisnost koja povezuje osnovne PrimeFaces biblioteke sa programom
- primefaces-extensions – za neke funkcionalnosti korišćene su PrimeFaces komponente koje nisu u standardnom paketu
- commons-email – obezbeđuje jednostavan API za slanje mejlova iz Java aplikacije
- guava-collections – pomoćna biblioteka korišćena za konverziju Java seta u listu
- primefaces.all-themes – za grafički prikaz korišćena je nestandarna primefaces tema iz ove biblioteke

U nastavku se nalazi slika celokupnog *pom.xml* fajla (**Error! Reference source not found., Error! Reference source not found.**).

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-email</artifactId>
  <version>1.5</version>
</dependency>
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava-collections</artifactId>
  <version>r03</version>
</dependency>
<dependency>
  <groupId>org.primefaces.extensions</groupId>
  <artifactId>all-themes</artifactId>
  <version>1.0.8</version>
  <type>pom</type>
</dependency>
</dependencies>
```

Slika 5.2.4

```

<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.14</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.3.1.Final</version>
  </dependency>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-agroal</artifactId>
    <version>5.3.11.Final</version>
    <type>pom</type>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-java8</artifactId>
    <version>5.1.0.Final</version>
  </dependency>
  <dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.11</version>
  </dependency>
  <dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>7.0</version>
  </dependency>
  <dependency>
    <groupId>org.primefaces.extensions</groupId>
    <artifactId>primefaces-extensions</artifactId>
    <version>7.0</version>
  </dependency>

```

Slika 5.2.5

5.3 Realizacija funkcionalnosti

Softversko rešenje veb aplikacije organizovano je po MVC arhitekturi koristeći JSF radni okvir. View deo arhitekture predstavljaju JSF veb stranice. Korisnik interaguje sa View-om, koji sve korisničke zahteve prosleđuje odgovarajućem kontroleru. Kontroleri su realizovani kao javax ManagedBean-ovi. Oni sadrže logiku za obrađivanje korisničkih zahteva, dok se za potrebne podatke iz baze obraćaju odgovarajućoj DAO (*Data Access Object*) klasi (napominje se klasi, ne objektu jer su sve metode ove klase statičke pa je potreba za kreiranjem objekata datom tipa eliminisana). Model deo arhitekture čine DAO klase za povezivanje sa bazom i klase u koje su mapirani entiteti iz baze. U nastavku će ovo biti demonstrirano konkretnim primerima koda.

Veb stranica za logovanje (Slika 5.3.1), kao i sve ostale.xhtml stranice u sistemu, koristi PrimeFaces komponente kao zamenu za standardne JSF komponente gde kod je to moguće. Razlog ovoga elegantniji interfejs, ali i činjenica da PrimeFaces nudi brojne pogodnosti kao što je *Ajax* osvežavanje prikaza. Na primeru se vidi na koji način se sadržaj stranice povezuje sa odgovarajućim poljem kontrolera zaduženog za upravljanje ovim radnjama. Nakon što korisnik unese tražene podatke i pritisne dugme potvrdi vrednosti polja za korisničko ime i lozinku objekta kontrolera biće postavljeni na unete vrednosti. Potom će se izvršiti se javna metoda LoginKontrolera koja vrši samo logovanje u sistem.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
  <h:head>
    <title>Baza recenzenata</title>
  </h:head>
  <h:body>
    <h:outputStylesheet library="css" name="dizajn.css"/>
    <div class="sadržaj">
      <h2>Login</h2>
      <h:form id="forma">
        <p:outputLabel id="poruka" styleClass="poruka" value="#{LoginKontroler.poruka}"/>
        <h:panelGrid columns="3">
          <p:outputLabel value="Korisničko ime: "/>
          <p:inputText id="korime" value="#{LoginKontroler.korime}" required="true" requiredMessage="Morate uneti korisničko ime!"/>
          <p:message for="korime"/>
          <p:outputLabel value="Lozinka: "/>
          <p:password id="lozinka" value="#{LoginKontroler.lozinka}" required="true" requiredMessage="Morate uneti lozinku!"/>
          <p:message for="lozinka"/>
        </h:panelGrid>
        <br/><br/>
        <p:commandButton value="Potvrdi" action="#{LoginKontroler.login()}" update="forma"/>
        <br/><br/>
        <p:commandLink value="Zaboravili ste lozinku?" action="zaboravljenaLozinka?faces-redirect=true"/>
        <br/><br/><br/>
        <p:commandLink value="Nazad" action="/index?faces-redirect=true"/>
      </h:form>
    </div>
  </h:body>
</html>
```

Slika 5.3.1

LoginKontroler klasa (Slika 5.3.2) je anotirana sa *ManagedBean* i *SessionScoped* iz *java.faces.bean* paketa. Time je rečeno da se objekat ove klase automatski i referencira iz JSF stranica imenom određenim name atributom *ManagedBean* anotacije. *SessionScoped* određuje životni vek tog objekta, koji je u ovom slučaju vezan za trajanje trenutne sesije. U metodi login demonstrirana je primena algoritma za heširanje *sha256* koji je korišćen za enkripciju lozinke radi sigurnog čuvanja u bazi. U slučaju neregularnih ulaznih podataka u polje poruka, koje je povezano sa *outputText* tagom gorenavedene veb stranice, čime je demonstrirana dvosmernost komunikacije između JSF stranica i kontrolera.

Još jedan važan aspekt aplikacije koji je demonstriran u datom kodu je korišćenje atributa sesije kao način komunikacije između kontrolera u sistemu. Opredelilo se za ovaj način realizacije komunikacije između nezavisnih kontrolera jer predstavlja fleksibilno i lako promenljivo rešenje. Pamćenje trenutnog korisnika značajno je omogućavanje brojnih daljih funkcionalnosti sistema.

```
@ManagedBean(name = "LoginKontroler")
@SessionScoped
public class LoginKontroler implements Serializable {

    private String korime;
    private String lozinka;

    private String poruka;

    public String login() {
        lozinka = DigestUtils.sha256Hex(lozinka);
        Korisnik korisnik = LoginDAO.dohvatiKorisnika(korime, lozinka);
        if (korisnik == null) {
            poruka = "Uneto korisnicko ime i lozinka nisu prepoznati!";
            PrimeFaces.current().ajax().update("forma:poruka");
            return "login";
        }
        HttpSession sesija = SessionUtils.getSession();
        sesija.setAttribute("korisnik", korisnik);
        poruka = null;
        switch (korime) {
            case "admin":
                sesija.setAttribute("admin", true);
                return "/faces/admin/admin?faces-redirect=true";
            default:
                sesija.setAttribute("recenzent", true);
                return "/faces/korisnik/pocetna?faces-redirect=true";
        }
    }

    public String logout() {
        HttpSession sesija = SessionUtils.getSession();
        sesija.invalidate();
        return "/index?faces-redirect=true";
    }
}
```

Slika 5.3.2

Kako bi se upotpunio prikaz vertikale sistema priložen je kod klase *LoginDAO* (Slika 5.3.3) kao predstavnik sloja za komunikaciju sa bazom. Za povezivanje sa bazom koristi se statički inicijalizovan objekat *SessionFactory* iz *org.hibernate* paketa. Pomoću njega se započinje i zatvara sesija za komunikaciju sa bazom, koja se vrši pomoću HQL upita. Ovaj pristup omogućava da se u upitima navode nazivi klasa i atributa klasa umesto naziva tabela i kolona iz baza. Takođe, podaci iz baze direktno se mapiraju u objekat odgovarajuće entitetske klase. Sve ovo znato pojednostavljuje kod i podiže nivo apstrakcije na kome programer razmišlja.

```
public class LoginDAO {

    private static final SessionFactory factory;

    static {
        Configuration configuration = new Configuration();
        configuration.configure();
        ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
            .applySettings(configuration.getProperties()).build();
        factory = configuration.buildSessionFactory(serviceRegistry);
    }

    public static Korisnik dohvatiKorisnika(String korisnickoIme, String lozinka) {
        Korisnik nadjen;
        Session sesija = factory.openSession();
        String HQL = "from Korisnik where korisnickoIme = :ime and lozinka = :lozinka";
        Query upit = sesija.createQuery(HQL);
        upit.setParameter("ime", korisnickoIme);
        upit.setParameter("lozinka", lozinka);
        nadjen = (Korisnik) upit.uniqueResult();
        sesija.close();
        return nadjen;
    }

    public static boolean korisnikPrihvacen(Korisnik k) {
        Korisnik nadjen;
        boolean rezultat;
        Session sesija = factory.openSession();
        String HQL = "from Korisnik where korisnickoIme = :ime and stanje = 'P'";
        Query upit = sesija.createQuery(HQL);
        upit.setParameter("ime", k.getKorisnickoIme());
        nadjen = (Korisnik) upit.uniqueResult();
        if (nadjen != null) {
            rezultat = nadjen.getStanje().equals("P");
        } else {
            rezultat = false;
        }
        sesija.close();
        return rezultat;
    }

    public static Korisnik dohvatiKorisnikaPoId(int id) {
        Korisnik k;
        Session sesija = factory.openSession();
        k = (Korisnik) sesija.get(Korisnik.class, id);
        sesija.close();
        return k;
    }
}
```

Slika 5.3.3

Nešto specifičniju strukturu ima stranica za ocenjivanje projekata od strane recenzenta (Slika 5.3.4). Kako je i broj pitanja i broj ponuđenih odgovora promenljiv u zavisnosti od konkretnog programskog poziva generisanje prikaza istih moralo se uradi na poseban način. Za generisanje pitanja korišćena je *ui:repeat* JSF komponenta koja generiše sadržaj naveden između početnog i krajnjeg taga onoliko puta koliko elemenata ima lista koja se odredi atributom *value*. Ponaša se kao petlja koja generiše sadržaj u zavisnosti od vrednosti trenutnog elementa liste, čiji je naziv definisan atributom *var*. Komponenta *f:selectItems* generiše po jedan checkbox za svaki element mape (parovi teksta koji se prikazuje i vrednosti samog polja) koja joj se prosledi u atributu *value*, čime je rešen problem promenljivog broja odgovora na pitanje.

```
<h:form id="forma">
  <ui:repeat value="#{ocenjivanjeKontroler.listaPitanja}" var="pitanje">
    <p:outputLabel value="#{pitanje.tekst}"/>
    <br/><br/>
    <p:selectManyCheckbox value="#{pitanje.tekstoviOdgovora}" layout="pageDirection">
      <f:selectItems value="#{ocenjivanjeKontroler.napraviMapuOdgovora(pitanje)}" />
    </p:selectManyCheckbox>
    <br/><br/>
  </ui:repeat>
  <br/><br/>
  <p:outputLabel value="Ocena projekta: "/>
  <p:inputText id="ocena" value="#{ocenjivanjeKontroler.ocena}"/><br/><br/>
  <br/><br/>
  <p:commandButton value="Sačuvaj sve" action="#{ocenjivanjeKontroler.samoSacuvaj()}" />
  <br/><br/>
  <p:commandButton value="Sačuvaj i zaključaj" action="#{ocenjivanjeKontroler.sacuvajIZaključaj()}" />
  <br/><br/><br/>
  <p:commandLink value="Nazad" action="/faces/korisnik/pocetna?faces-redirect=true"/>
</h:form>
```

Slika 5.3.4

U slučaju da je recenzent već odgovarao na pitanja ali ih nije zaključao pa želi da promeni odgovore, *OcenjivanjeKontroler* pri inicijalizaciji ima zadatak da iz baze učita te odgovore i mapira u format adekvatan za prosleđivanje veb stranici. Kod metode za inicijalizaciju datog kontrolera priložen je u nastavku (Slika 5.3.5). Iz atributa sesije se dobijaju objekti Korisnika i Projekta dobija se objekat klase *KorisnikProjekat* pomoću kog se dobija spisak svih odgovora koje je dati korisnik dao. Korišćenjem pogodne *removeIf()* metode iz kolekcije se izbacuju svi odgovori koji ne pripadaju datom projektu. Nakon toga se elementi mape ubacuju u niz stringova jer je to forma koju PrimeFaces komponenta *p:selectManyCheckbox* prepoznaje i očekuje. Metoda *napraviMapuOdgovora* transformiše listu stringova u mapu čija je upotreba objašnjena u prethodnom pasusu.

```

@PostConstruct
public void inicijalizacija() {
    projekat = (Projekat) SessionUtils.getSession().getAttribute("projekat");
    korisnik = (Korisnik) SessionUtils.getSession().getAttribute("korisnik");
    KorisnikProjekat korisnikProjekat = KorisnikProjekatDAO.nadjiPoKorisnikuIProjektu(korisnik, projekat);
    ProgramskiPoziv programskiPoziv = projekat.getProgramskiPoziv();
    Set<Pitanje> pitanja = programskiPoziv.getPitanjes();
    listaPitanja = Lists.newArrayList(pitanja);

    if (korisnikProjekat.getStanjeOdgovora().equals("sacuvani")) {
        for (Pitanje pitanje : listaPitanja) {
            korisnikProjekat = KorisnikProjekatDAO.nadjiPoKorisnikuIProjektu(korisnik, projekat);
            Set<KorisnikOdgovor> korisnikOdgovori = korisnikProjekat.getKorisnikOdgovors();
            korisnikOdgovori.removeIf(((KorisnikOdgovor ko) -> !Objects.equals(ko.getPitanje().getIdpitanje(), pitanje.getIdpitanje())));
            Iterator<KorisnikOdgovor> iterator = korisnikOdgovori.iterator();
            String[] tekstoviOdgovora = new String[korisnikOdgovori.size()];
            int indeks = 0;
            while (iterator.hasNext()) {
                tekstoviOdgovora[indeks++] = iterator.next().getOdgovor().getTekst();
            }
            pitanje.setTekstoviOdgovora(tekstoviOdgovora);
        }
        ocena = korisnikProjekat.getOcena();
    }
}

```

Slika 5.3.5

5.4 Problemi prilikom realizacije sistema

Kao i pri razvoju svakog projekta, prvi problem sa kojim se susrelo pri razvoju ovog je adekvatan odabir tehnologija i alata koji će se koristiti. Korišćenje Java programskog jezika prirodno se nametnulo s obzirom da je reč o veb aplikaciji, dok je korišćenje JSF radnog okvira u startu eliminisalo brojne potencijalne probleme oko komunikacije frontend-a i backend-a.

Jedan od ključnih elemenata koji su doprineli uspešnosti i preventovali niz potencijalnih problema pri izradi projekta bilo je korišćenje Maven-a. Bilo koja iole komplikovanija aplikacija bila bi višestruko teža za realizaciju bez pomoći alata kao što je Maven. Ne samo da bi programer morao manuelno da povezuje svaki jar fajl u sistem, već bi morao da razmišlja o problemima kompatibilnost i verzija eksternih biblioteka, dok se pomoću Maven-a sve to rešava kroz svega par linija koda u pom.xml.

Odlučeno je da se komunikacija sa bazom obavlja pomoću Hibernate API-ja kako bi se maksimalno automatizovao i ubrzao proces razvoja tog dela aplikacije. Pogodnosti razvojnog okruženja kao što automatski ORM i automatsko generisanje konfiguracionih fajlova, dodatno su olakšali razvoj ovog inače prilično kompleksnog dela sistema. Bez ovog sloja aplikacije, komunikacija sa bazom bila bi znatno složenija i opterećena detaljima, što neizostavno ostavlja prostor za nepredviđene greške. Ovo je još jedan primer problema koji su u startu preventovani izborom adekvatne tehnologije

Naravno, uprkos svemu tokom procesa razvoja ovog softvera nailazilo se na određene tehničke probleme koji su se pokušali prevazići na najbolji način. Nekoliko primera tih problema biće izloženi u nastavku.

Jedan od nešto specifičnijih korisničkih zahteva odnosio se na mogućnost slanja poruke putem elektronske pošte od strane same aplikacije. Java programski jezik ima Java Mail API koji je inicijalno bio nameravan da se koristi u ovoj aplikaciji. Utvrđeno je, međutim, da on zahteva parametrizaciju brojnih faktora koji nisu od naročitog interesa za potrebe datog sistema. Iz tog razloga se odlučilo za korišćenje Commons Email API-ja, koji predstavlja nagodradnju na Java Mail API i samim tim ga uprošćava. Kod pomoću koga se realizovala tražena funkcionalnost priložen je u nastavku (Slika 5.4.1).

```

public static void posaljiMejl(String primalac, String naslov, String telo) {
    try {
        Email email = new SimpleEmail();
        email.setHostName("smtp.gmail.com");
        email.setSmtpPort(25);
        email.setAuthenticator(new DefaultAuthenticator("bazarecenzenata@gmail.com", "bazarecenzenata123!"));
        email.setFrom("bazarecenzenata@gmail.com");
        email.setSubject(naslov);
        email.setMsg(telo);
        email.addTo(primalac);
        email.setStartTLSRequired(true);
        email.send();
    } catch (EmailException ex) {
        Logger.getLogger(SlanjeMejla.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Slika 5.4.1

Još jedan od problema pri realizaciji sistema o kome se nešto duže diskutovalo bio je način čuvanja tekastualnih fajlova, kako biografija recenzenata, tako i dokumenata projekata. Postojala je dilema da li čuvati fajlove u samoj bazi podataka ili u fajl sistemu servera. Nakon kraćeg istraživanja utvrđeno je da čuvanje podataka u fajl sistemu ima brojne prednosti u odnosu na čuvanje podataka u bazi. Naime, memorija fajl sistema često je jeftinija od memorije u bazi, iz Java aplikacije se može direktno pristupi fajl sistemu, što sa bazom podataka nije slučaj, brzina pristupa fajl sistemu ima brojne načine optimizacije koristeći pomoćne aplikacije, što opet sa bazom nije slučaj. Iz ovih razloga odlučeno je da se svi fajlovi čuvaju u posebnom folderu fajl sistema, dok se u bazi čuva samo lokacija i naziv dokumenata. Radi jedinstvenosti imena svakog fajla pribeglo se automatskom generisanju drugog dela imena fajla. Deo koda zadužen za čuvanje učitanoog dokumenta u fajl sistemu priložen je u nastavku (Slika 5.4.2).

Učitavanje dokumenata sa frontenda omogućeno je korišćenjem PrimeFaces taga *p:fileUpload* u kome se specificira u koji objekat tipa *UploadedFile* iz *org.primefaces.model* paketa se želi učitati uneti dokument, zatim da li se želi omogućiti učitavanje samo jednog fajla, ili više njih, kao i brojne druge pomoćne funkcionalnosti.

Sa druge strane, prikaz učitanoog fajl, jednostavno se vršio otvaranjem linka sa odgovarajućom putanjom do fajla u novom prozoru ili u okviru *iframe* taga.

```

public void sacuvajFajlJSF(FileUploadEvent e) {
    fajl = e.getFile();
    sacuvajFajl(fajl);
}

private String sacuvajFajl(UploadedFile fajl) {
    String konacnoImeFajla = null;
    try (InputStream input = fajl.getInputStream()) {
        String putanja = putanjaZaCuvanje + nazivFoldera;
        Path folder = Paths.get(putanja);
        if (!Files.exists(folder)) {
            try {
                Files.createDirectories(folder);
            } catch (IOException ex) {
                Logger.getLogger(RegistracijaKontroler.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        String imeFajla = FilenameUtils.getBaseName(fajl.getFileName());
        String ekstenzija = FilenameUtils.getExtension(fajl.getFileName());
        Path temp = Files.createTempFile(folder, imeFajla + "-", "." + ekstenzija);
        Files.copy(input, temp, StandardCopyOption.REPLACE_EXISTING);
        konacnoImeFajla = temp.getFileName().toString(); // ***
    } catch (IOException ex) {
        Logger.getLogger(RegistracijaKontroler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return konacnoImeFajla;
}

```

Slika 5.4.2

Još jedan od problema sa kojim se susrelo tokom realizacije traženog sistema odnosio se na obezbeđivanje korektnog unosa naziva Programskog poziva pri kreiranju projekta, naziva projekta i rezenzenta prilikom dodeljivanja, odnosno oduzimanja, projekta recenzentu. Naime, u svim navedenim slučajevima trebalo je obezbediti da se podatak koji se želi referencirati zaista nalazi u sistemu. To je obezbeđeno u samom kod JSF stranica korišćenjem PrimeFaces taga *p:autoComplete* (Slika 5.4.3) koji ima za zadatak da pri korisničkom unosu, uz pomoć metode koja vraća listu elemenata tog tima iz baze čiji nazivi počinju unetim slovima (Slika 5.4.4), predloži korisniku u vidu padajuće liste odgovarajući naziv. Postavljajući odgovarajući atribut, onemogućava se unos naziva koji nije bio u listi ponuđenih čime se garantuje ispravan rad sistema.

```

<h:outputLabel value="Naziv projekta:"/>
<p:autoComplete forceSelection="true" value="#{projekatKontroler.unetProjekat}"
    completeMethod="#{projekatKontroler.predloziNazivProjekta}"/>

```

Slika 5.4.3

```

public List<String> predloziNazivProjekta(String zadatak) {
    List<String> rezultat = new ArrayList<>();
    for (Projekat projekat : projekti) {
        if (projekat.getNaziv().startsWith(zadatak)) {
            rezultat.add(projekat.getNaziv());
        }
    }
    return rezultat;
}

```

Slika 5.4.4

6 Zaključak

U svetu u kome je broj veb stranica i portala skoro neograničen teško je napraviti nešto novo što ima moć da zadovolji korisnika u potpunosti. Ovde se pokušalo napraviti softversko rešenje za specifičan problem za koji možda trenutno ne postoji dovoljno dobro rešenje u ponudi, sa jedne strane ocenjivanje naučnih projekata od strane ovlašćenih recenzenata, a sa druge upravljanje sistemom i praćenje rada recenzenata. Iako je softversko rešenje u trenutnoj formi daleko od spremnog za komercijalno tržište, smatra se da predstavlja dobru osnovu za dalji razvoj.

Prva mogućnost unapređenja vidi se u poboljšanju korisničkog interfejsa, koji je danas praktično jednako važan koliko i same funkcionalnosti. Upravo je to razlog zašto je odlučeno da se pored JSF radnog okvira pri razvoju koristi i PrimeFaces biblioteka. Aspekt u kom se vidi mogućnost poboljšanja je organizacija samog interfejsa, u smislu organizacije stranica, naziva podсистема i slično. Posao pravljenja optimalnog i intuitivnog korisničkog interfejsa nije ni malo lak posao. Iako se ovde pokušalo doći do što boljeg rešenja u datim okolnostima, smatra se da prostor za napredak svakako postoji.

Sledeće na čemu bi trebalo da se radi jeste proširenje skupa funkcionalnosti, kako na strani recenzenta, tako i na strani administratora. Kao prvo, administrator bi trebalo da ima mogućnost komunikacije sa recenzentima pojedinačno radi, na primer, dobijanja dodatnih informacija o oceni projekta koje bi mogle da pomognu pri donošenju konačne odluke. Ova funkcionalnost mogla bi vrlo lako da se implementira koristeći već postojeći kod za slanje poruka na adresu elektronske pošte recenzenta. Međutim, kako bi administrator mogao da dobije odgovor na poruku morao bi se implementirati novi deo sistema koji bi prevazišao problem prijema mejla od strane aplikacije.

Mogućnosti za proširenje skupa funkcionalnosti ima uvek i ima na pretek. Ovde je izdvojeno nekoliko koje se vide kao mogući prvi koraci na putu optimizacije i daljeg razvoja aplikacije.

Cilj ovog korisničkog uputstva bilo je detaljno upoznavanje čitaoca sa datim softverskim rešenjem, kako sa motivacijom za odabir baš ove teme, tako i sa korišćenim tehnologijama. Na početku je izložen cilj aplikacije sa detaljnim pregledom korisničkih zahteva. Zatim su ukratko opisane korišćene tehnologije, što je za cilj imalo bolje razumevanje zašto se baš za njih odlučilo pri izradi projekta. Opis rada sistema nudi detaljno i slikovito korisničko uputstvo za korišćenje aplikacije. U narednom delu objašnjen je način implementacije sistema za koji se odlučilo, sa pregledom delova koda za koje je autor smatrao da su od najvećeg interesa. Zatim su čitaocu izloženi problemi sa kojima se nailazilo tokom razvoja, uz priloženja rešenja istih za koje se opredelilo, kao i načini na koje su određeni problemi preventovani. Na kraju se diskutovalo o budućnosti sistema, mogućnosti izmena postojećeg rešenja i nadogradnja na isto, uz osvrt na obrađene teme.

7 Literatura

- Schildt, H. (2014). Java: A Beginner's Guide, Sixth Edition. Publisher: McGraw-Hill Education
- Oracle (2016). JavaFaces Technology. Preuzeto 11. decembra 2019, sa <https://www.oracle.com/technetwork/java/javase/javaserverfaces-139869.html>
- Hibernate (nd). Hibernate. Preuzeto 11. decembra 2019, sa <http://hibernate.org/>
- Kraus, L. (2015). Programski jezik Java sa rešenim zadacima. Beograd: Akademska misao
- The Apache Software Foundation (2002). Maven. Preuzeto 11. decembra 2019, sa <https://maven.apache.org/>
- Tutorials Point (2006). JPA Tutorial. Preuzeto 11. decembra 2019, sa <https://www.tutorialspoint.com/jpa/>
- Bauer, C., King, G., i Gregory, G. (2015). Java Persistence with Hibernate, Second Edition. NewYork: Manning publications
- Apache Commons (2018). Apache Commons Email. Preuzeto 11. decembra 2019, sa <https://commons.apache.org/proper/commons-email/>
- PrimeFaces (nd). PrimeFaces. Preuzeto 11. decembra, sa <https://www.primefaces.org/>
- Developer Mozilla (nd). Ajax – Web Developer's Guide. Preuzeto 11. decembra 2019, sa <https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX>