Elektrotehnički fakultet, Univerzitet u Beogradu

**A close up of a sign

Description automatically generated**

**Diplomski rad**

**Softverski sistem za povezivanje lovaca**

Kandidat: Aleksa Petrović

Mentor: prof. dr Boško Nikolić

29. Septembar 2020.

Sadržaj

[1. Uvod 3](#_Toc52027026)

[2. Opis zahteva sistema 5](#_Toc52027027)

[3. Opis korišćenih tehnologija 7](#_Toc52027028)

[3.1. Netbeans 7](#_Toc52027029)

[3.2. Java Database Connectivity 7](#_Toc52027030)

[3.3. MySQL 7](#_Toc52027031)

[3.4. Microsoft MySQL Workbench 8](#_Toc52027032)

[3.5. AES algoritam 8](#_Toc52027033)

[3.6. JSF 8](#_Toc52027034)

[3.7. CSS 8](#_Toc52027035)

[3.8. Facelets 9](#_Toc52027036)

[3.9. AJAX 9](#_Toc52027037)

[3.10. PrimeFaces 9](#_Toc52027038)

[3.11. OmniFaces 10](#_Toc52027039)

[3.12. JavaScript 10](#_Toc52027040)

[3.13. Java Message Service 10](#_Toc52027041)

[4. Opis rada sistema 11](#_Toc52027042)

[4.1. Naslovna stranica 11](#_Toc52027043)

[4.2. Registracija 11](#_Toc52027044)

[4.2.1. Unos podataka 11](#_Toc52027045)

[4.2.2. Potvrda podataka 13](#_Toc52027046)

[4.3. Zaboravljena lozinka 14](#_Toc52027047)

[4.4. Prijava 14](#_Toc52027048)

[4.5. Početna stranica 15](#_Toc52027049)

[4.5.1. Navigacioni meni 15](#_Toc52027050)

[4.5.2. Podaci korisnika lovac 16](#_Toc52027051)

[4.5.3. Podaci korisnika lovačko društvo 17](#_Toc52027052)

[4.5.4. Galerija slika 18](#_Toc52027053)

[4.5.5. Vesti i objave 18](#_Toc52027054)

[4.5.6. Objavljivanje vesti 19](#_Toc52027055)

[4.5.7. Pretraživanje korisnika 20](#_Toc52027056)

[4.6. Administratorska stranica 21](#_Toc52027057)

[4.7. Stranica gost 22](#_Toc52027058)

[5. Realizacija sistema 23](#_Toc52027059)

[5.1. Serverska strana aplikacije 24](#_Toc52027060)

[5.1.1. Modeli 24](#_Toc52027061)

[5.1.2. Kontroleri 24](#_Toc52027062)

[5.1.3. Paket Util 25](#_Toc52027063)

[5.1.4. Paket Util.dao 26](#_Toc52027064)

[5.2. Klijentska strana 26](#_Toc52027065)

[5.3. Povezivanje sa bazom 27](#_Toc52027066)

[5.4. Delovi koda implementacije sistema 29](#_Toc52027067)

[5.4.1. Prijava, registracija i verifikacija 29](#_Toc52027068)

[5.4.2. Gmail service 33](#_Toc52027069)

[5.4.3. Prikazivanje vesti 35](#_Toc52027070)

[5.4.4. Pretraživanje korisnika 41](#_Toc52027071)

[6. Zaključak 44](#_Toc52027072)

[7. Literatura 46](#_Toc52027073)

# 1. Uvod

Tim Berners-Lee je čovek koji je 1989. godine osmislio World Wide Web (WWW), danas poznatiji kao internet. Nedugo zatim, do oktobra 1990. godine, precizirao je tri osnovne tehnologije koje su temelj i današnjeg web-a: HTML, URI i HTTP[1]. Tokom prvih nekoliko godina od njihovog predstavljanja HTML stranice su bile statičke. Do 1995. godine bilo je dosta unapređenja, ali najveće od svih je bilo predstavljanje JavaScript jezika. Predstavljanjem JavaScript jezika stranice postaju dinamičnije. Serveri više nisu morali da generišu cele stranice, već su to činile ugrađene skripte, što je internet učinilo dosta bržim[2]. Javno predstavljanje AJAX-a 2005. godine, koji je prvi put korišćen još 1999. godine, bila je nova velika revolucija u web programiranju. To nije bila nova tehnologija, već kombinacija grupe već poznatih tehnologija. Predstavljanjem AJAX-a dobijen je responsivan korisnički interfejs (UI). Klijentska web stanica više nije morala da čeka odgovor servera već je korisnik bio u mogućnosti da i dalje koristi web stranicu. Čim server odgovori, relevantna funkcija manipulisala je podacima u pozadini[3]. Naposletku, razvoj HTML5 tehnologije, koja je zvanično prihvaćena kao standard 2014. godine, doneo je nekoliko novih elemenata i atributa koje danas koriste moderni sajtovi.

Web sajt je definisan njegovim sadržajem, dok je web aplikacija definisana samom interakcijom sa korisnikom[4]. Web sajt je grupa globalno dostupnih, međusobno povezanih web stranica koje imaju jedan domen.Web aplikacija je softer koji radi u pozadini, na serveru, koristeći bilo koji pretraživač[5]. Od moderne web aplikacije se očekuje da komunicira sa korisnicima i ispunjava razne njihove zahteve. U poređenju sa desktop aplikacijama, web aplikacija se može koristiti na svim platformama (Windows, Linux, Mac...). Dostupna je u bilo kom trenutku i lako ju je održavati, s obzirom da je internet dostupan svuda.

Cilj diplomskog rada je upravo kreiranje jedne moderne web aplikacije, kreiranje sistema za povezivanje lovaca. U sistemu postoje tri vrste korisnika, svaki od njih sa posebnim privilegijama. U daljem tekstu biće prikazani alati i tehnologije korišćeni kako bi se realizovala ovakva jedna aplikacija. Kao razvojno okruženje korišćen je NetBeans. Razvojno okruženje pruža podršku za JSF i Java Enterprise edition (Java EE), takođe uključuje i GlassFish server. Za modelovanje baze podataka korišćen je Microsoft MySQL Workbench.

Serverska strana aplikacije realizovana je kroz programski jezik Java, uz korišćenje JSF anotacija i korićenje javax i activation biblioteke za slanje mejlova. Komunikacija sa bazom omogućena je korišćenjem mySQL connector-a. Klijentska strana aplikacije je implementirana kroz JSF, korišćenjem biblioteka PrimeFaces i OmniFaces i upotrebom AJAX tehnologije za osvežavanje stranice.

Aplikacija ima za cilj prikupljanje i razmenu podataka o bivšim i sadašnjim lovcima, kao i uspostavljanje njihovih medjusobnih veza. Korisnici mogu deliti svoje privatne informacije, ili ih menjati. Postoji mogućnost objavljivanja vesti i pitanja, kao i privatnih fotografija. Vesti su podeljene u kategorije koje određuje sam administrator. Moguće je postavljanje privatnosti na objavljenu vest, pravljenje privatnih grupa gde korisnici mogu objavljivati zanimljivosti vezane za datu grupu korisnika. Aplikacija podržava pretragu korisnika pod njihovim imenom i prezimenom. Korisnici koji nemaju nalog na portalu, mogu se prijaviti kao gosti. Gosti portala imaju pristup vestima označenim kao javne, kao i pristup informacijama vezanim za administratore portala. Modul administratora realizovan je na način da omogući upravljanje celokupnom aplikacijom. Administrator je u mogućnosti da obriše uvredljive objave, definiše nove vrednosti šifrarnika za kategoriju vesti, odgovara na pitanja korisnika. Iako postoji pregršt društvenih mreža za povezivanje ljudi poput Facebook-a, sam sistem za povezivanje lovaca i lovačkih društava je aplikacija namenjena ljudima i organizacijama orjentisanim ka određenoj delatnosti, lovu. U daljem tekstu biće detaljno opisane tehnologije i alati korišćeni za realizaciju aplikacije, opis rada same aplikacije, realizacija samog sistema uz pregled i opis najbitnijih delova koda.

# 2. Opis zahteva sistema

Web aplikacija mora da omogući što brži odziv samom korisniku. Korisnički interfejs mora biti responsivan i razumljiv samom korisniku. Polja za unos su validirana određenim regex izrazima. Posebna pažnja posvećena je samoj registraciji korisnika. Korisnik je obavezan da unese tačne podatke i verifikuje ih putem mejl adrese kako bi mogao da se prijavi na sistem.

Web aplikacija mora da ispuni sledeće funkcionalnosti:

Funkcionalnosti portala podeljene su u kategorije prema vrstama korisnika na sledeći način:

1. Osnovne funkcionalnosti
2. Opšte funkcionalnosti
3. Funkcionalnosti dostupne lovcima i lovačkim društvima
4. Funkcionalnosti dostupne administratoru

Funkcionalnosti dostupne gostima portala – Osnovne funkcionalnosti:

1. Pregled osnovnih podataka o portalu
2. Pregled vesti koje su označene kao javne
3. Kontaktiranje administratora
4. Mogućnost prijavljivanja
5. Mogućnost registracije

Funkcionalnosti koje su dostupne registrovanim korisnicima portala – Opšte funkcionalnosti:

1. Postavljanje objave
2. Postavljanje pitanja
3. Kreiranje grupa
4. Pretraga lovaca i lovačkih društava

Funkcionalnosti dostupne lovcima i lovačkim društvima:

1. Pregled i ažuriranje profila

Funckionalnosti dostupne administratoru:

1. Ažuriranje kategorija vesti
2. Ažuriranje šifrarnika
3. Brisanje neadekvatnog sadržaja
4. Postavljanje reklamnog materijala
5. Izveštaji
6. Komunikacija sa korisnicima portala

# 3. Opis korišćenih tehnologija

Opis tehnologija i alata korišćenih za realizaciju sistema.

## 3.1. Netbeans

Netbeans je integrisano razvojno okruženje prvenstveno namenjeno Java tehnologijama, ali takođe pruža podršku i ostalim programskim jezicima kao što su C, C++, Python, Rubi i drugi. Netbeans radi na različitim platformama kao što su Windows, Linux, Apple. Podržava različite tehnologije i alate koji poboljšavaju razvojni proces aplikacije[6]. Verzija Netbeans-a korišćena pri izradi ovog sistema je 8.2, kao i GlassFish server 4.1.

## 3.2. Java Database Connectivity

JDBC je API za programski jezik Java, koji definiše način kako klijent može da pristupi bazi podataka. JDBC omugućava izvršavanje četiri osnovne operacije(CRUD):

* Pravljenje nove tabele(Create)
* Ubacivanje podataka u tabelu(Insert)
* Ažuriranje tabele(Update)
* Brisanje podataka iz tabele(Delete)

JDBC upiti se izvršavaju koristeći klase kao što su Statement, PreparedStatement, CallableStatement, itd[7].

## 3.3. MySQL

MySql je open source sistem upravljanja relacionom bazom podataka(RDBMS). Koristi Structured Query Language. MySQL se može koristiti na različitim platformama i u različitim programskim jezicima[8].

## 3.4. Microsoft MySQL Workbench

MySQL Workbench je besplatni alat za kreiranje baze podataka. On omogućava modeliranje same baze podataka, pisanje SQL upita i sveobuhvatne administrativne alate za konfiguraciju servera, administraciju korisnika, izradu sigurnosnih kopija i još mnogo toga. Izmedju ostalog, moguće je pisati trigger funkcije za validaciju unetih podataka. Dostupan je na raznim platformama[8].

## 3.5. AES algoritam

Advanced Encryption Standard(AES), takođe poznat i pod originalnim imenom Rijndael je specifikacija za enkripciju podataka. Podržava različite veličine ključeva 128, 192 i 256 bitova. MySQL pruža podršku za korišćenje AES algoritma, funkcije AES\_ENCRYPT(key, value) i AES\_DECRYPT(key, value)[9]. U realizaciji aplikacije korišćen je ključ sa 128 bitova.

## 3.6. JSF

Java Server Faces(JSF) je korisnički interfejs za platformu Java EE. To je zapravo web framework baziran na Javi. JSF je uključen u Java EE platformu, tako da nije potrebno dodavati dodatne biblioteke u projekat. JSF 2.x koristi Facelets kao njegov podrazumevani templating[10]. JSF nije samo običan web framework, on nudi ekosistem prenosivih UI komponenti i biblioteka[11]. To znači da možete koristiti neke druge komponente, tako što jednostavno ubacite odgovarajuću biblioteku u projekat. JSF pruža podršku za AJAX, što je od velike važnosti, pošto se podrazumeva da današnje stranice budu što dinamičnije, kao i da odziv bude najbrži moguć.

## 3.7. CSS

Kaskadna tabela stilova(CSS) je jezik za stilizovanje tabela i ostalih elemenata jezika HTML. On je temelj WWW-a, isto kao što su to HTML i JavaScript[2]. Dizajniran je tako da omogućava drugačiji izgled određenih elemenata HTML-a, uključujuči i njihove boje, fontove, njihovu tranziciju na web stranici, itd. Pomoću selektora bira se ciljani element i izmene se primenjuju samo na njega. Pored selektora postoje i ugrađene klase, koje se odnose na svaki element određenog tipa.

## 3.8. Facelets

Facelets je jako moćan jezik, koji se koristi za izgradnju prikaza JSF stranica koristeći HTML stil za izgradnju stabala komponenti[12]. Karakteristike Facelets-a uključuju sledeće:

* Korišćenje XHTML-a za kreiranje web stranica
* Podrška za Facelets tagove pored JSF tagova na samoj web stranici
* Podrška za jezik izražavanja (Expression Language)
* Templating za komponente i stranice

Templaiting predstavlja način kako se programski kod može iskoristiti što efikasnije, isti kod se može koristiti više puta na istoj web stranici. To je omogućilo da Facelets zvanično postane neizostavni deo JSF tehnologij počev od verzije 2.0.

## 3.9. AJAX

Asinhron JavaScript i XML(AJAX) je set tehnika koje se koriste u mnogim web tehnologijama na klijentskoj strani kako bi web aplikaciju učinilo što dinamičnijom[3]. Pojava AJAX-a je zapravo bila revolucija u programiranju, s obzirom na to da korisnik više nije morao da čeka odgovor servera, već je mogao da nastavi da pretražuje na web stranici. AJAX je podržan od strane mnogih biblioteka, kao što su JSF, PrimeFaces, OmniFaces, BootsFaces itd. U današnjim modernim aplikacijama neizostavan je deo njihove implementacije, s obzirom na to da ne zahteva ponovno učitavanje stranice, već se sve dešava u pozadini. Odziv web aplikacija je primenom seta tehnika AJAX-a postao dosta brži za razliku od statičkih web stranica, što je zapravo jedan od glavnih ciljeva web aplikacije.

## 3.10. PrimeFaces

PrimeFaces je open source korisnički interfejs, čija je biblioteka komponenata bazirana na JSF komponentama. Postao je neizostavan deo implementacije klijentske strane. Podržava jako bitne segmente, kao što su dijalozi (Dialog Framework) i validacija polja (Client Side Validation)[13]. Takođe podržava AJAX i doveo je do velikog napretka u vidu korišćenja JSF framework-a u tehnologijama koje nisu vezane samo za desktop web aplikacije.

## 3.11. OmniFaces

OmniFaces je open source uslužna biblioteka, namenjena za JSF 2.x framework[14]. Razvijena je pomoću JSF API-a, a cilj je da olakša funkcionalnosti JSF-a. OmniFaces je predstavljen od strane članova JSF ekspertske grupe (JSF EG), kao odgovor na brojna pitanja i probleme koje JSF nije mogao da reši. OmniFaces nudi jako dobru podršku za dinamično prikazivanje slika iz baze podataka.

## 3.12. JavaScript

JavaScript je dinamički jezik za programiranje. Jako je intuitivan i najčešće se koristi kao deo web stranica. Primena JavaScript jezika klijentskoj skripti omogućava interakciju sa korisnikom i izradu dinamičnih stranica. JavaScript spada u interpretirane jezike i poseduje neke od mogućnosti koje poseduju objektno orjentisani programski jezici. Najčešće se JavaScript izvršava na korisnikovom web pregledaču, što dodatno daje brži odziv jer nema komunikacije između klijentskog i serverskog dela aplikacije.

## 3.13. Java Message Service

Java Message Service (JMS) API je standard koji omogućava aplikacijama zasnovanim na Java EE pisanje, čitanje, slanje i primanje poruka[15]. Omogućava distribuiranu komunikaciju koja je labavo povezana (loosly coupled), pouzdana i asinhrona. Da bi se koristile usluge Message Service-a neophodno je uključiti sledeće biblioteke u projekat: javax i activation.

# 4. Opis rada sistema

## 4.1. Naslovna stranica

A person that is standing in the grass

Description automatically generated Prva stranica koja se pojavljuje pri podizanju servera je naslovna stranica. Kreativna slika je tu kako bi zainteresovala gosta portala da napravi svoj nalog. U navigacionom meniju nalaze se tri dugmeta: ,,Prijava“, ,,Registracija“ i ,,Gost“.

slika 1: Naslovna stranica

Klikom na dugme ,,Registracija“, pojavljuje se forma za registraciju korisnika. Nalog na portalu mogu napraviti samo lovci, dok su lovačka društva u obavezi da kontaktiraju administratora, čije podatke i kontakt mogu dobiti klikom na dugme gost.

## 4.2. Registracija

### 4.2.1. Unos podataka

Prvi prozor (slika 2) koji se pojavljuje traži od korisnika da unese svoje osnovne podatke: ime, prezime, srednje ime i datum rodjenja. Korisnik mora imati više od 18 godina kako bi stekao pravo da napravi korisnički nalog. Neophodno je popuniti sva polja. Klikom na dugme ,,Dalje“ prelazite na sledeći deo registracije, gde je neophodno uneti mesto i državu.

A close up of a sign

Description automatically generatedGraphical user interface, website

Description automatically generated

slika 3: Forma registracija, prozor 2

slika 2: Forma registracija, prozor 1

Graphical user interface, website

Description automatically generatedA close up of a sign

Description automatically generatedOpciono, možete uneti lovačko društvo i broj članske karte. Status lovca je podrazumevano neaktivan. Pritiskom na dugme ,,Dalje“, pojavljuje se prozor, kao na slici 5, na kome je neophodno uneti kontakt podatke: email i broj mobilnog telefona. Svako polje sistema je validirano odgovarajućim regex izrazom. Za polja koja je obavezno uneti, korisniku se ispisuje određena poruka ako ih ne unese. Ukoliko korisnik unese neodgovarajući format, ili u ovom slučaju umesto brojeva unese slova ili nešto drugo, ispisuje se odgovarajuča validaciona poruka.

slika 5: Forma registracija, prozor 4

slika 4: Forma registracija, prozor 3

### 4.2.2. Potvrda podataka

Graphical user interface, website

Description automatically generatedA picture containing text, baseball, person, holding

Description automatically generated Klikom na dugme ,,Potvrdi“ korisnik se preusmerava na novu stranicu gde mu se prikazuju svi uneti podaci. Ukoliko korisnik uoči neku nepravilnost koju je napravio prilikom unošenja podataka korisnik može da se vrati na prethodnu stranu klikom na link u gornjem levom uglu. Pošto korisnik utvrdi da su svi podaci koje je uneo tačni, klikom na link ,,Napravi Lozinku“ pojavljuje se nova forma za pravljenje lozinke.

slika 6: Forma za potvrdu unetih podataka

slika 7: Froma za pravljenje lozinke

Nova forma očekuje od korisnika da unese jedinstveno korisničko ime i napravi svoju lozinku. Ukoliko se vrednosti polja ,,Lozinka“ i ,,Potvrda lozinke“ ne poklapaju korisniku se u dijalogu ispisuje odgovarajuća poruka i forma se restartuje. Takođe u slučaju da već postoji korisnik sa istom e-mail adresom, korisniku će biti ispisana odgovarajuća poruka.

## 4.3. Zaboravljena lozinka

A picture containing person, standing, people, person

Description automatically generatedUkoliko je korisnik zaboravio svoje korisničko ime ili lozinku, klikom na dugme ,,Prijava“ a potom na link ,,Zaboravljena lozinka“ korisnik će biti preusmeren na novu stranicu. Na stranici se pojavljuje forma, gde se od korisnika zahteva da unese svoju e-mail adresu. Ukoliko adresa korisnika postoji u sistemu, korisniku će putem mejla biti poslato njegovo korisničko ime i lozinka. U suprotnom, korisniku se ispisuje odgovarajuća poruka.

slika 8: Stranica za restartovanje lozinke

## 4.4. Prijava

Klikom na dugme ,,Prijava“ korisniku se pojavljuje forma za unos podataka. Unosom korisničkog imena i lozinke korisnik se preusmerava na odgovarajuću stranicu, u zavisnosti od vrste korisnika.

Ukoliko korisnik nije verifikovao svoj nalog klikom na link koji je dobio u mejlu, poslatom na njegovu e-mail adresu koju je uneo tokom registracije, korisniku se neće biti omogućen pristup sajtu. Verifikacioni link traje tri dana, nakon čega se kosnički podaci brišu iz baze podataka i korisnik će morati da napravi novi nalog kako bi sajt mogao da koristi sa privilegijama lovca.

**Graphical user interface, diagram

Description automatically generated**

slika 9: Forma za prijavljivanje

## 4.5. Početna stranica

### 4.5.1. Navigacioni meni

A screenshot of a social media post

Description automatically generated Nakon uspešne autentifikacije, korisnik se preusmerava na početnu stranicu. U navigacionom meniju se nalazi dugme ,,Početna Strana“ koje osvežava početnu stranu. Pored njega, nalazi se polje i dugme ,,Pretraži“, koji služe za pretraživanje korisnika sistema.

slika 10: Početna stranica

Pretraživanje se vrši unošenjem početka imena, ili celog imena korisnika, i početka prezimena. Klikom na dugme ,,Pretrazi“ korisniku će se na desnoj polovini početne strane prikazati rezultati pretrage. ,,Odjava“ dugme, koje se nalazi na desnoj strani navigacionog menija, korisnika vraća na naslovnu stranu aplikacije.

### 4.5.2. Podaci korisnika lovac

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated Korisnik lovac može videti svoje podatke na levoj polovini početne strane. Prikazani podaci se mogu izmeniti pritiskom na dugme ,,Izmeni“. Korisnik podešava svoj profil tek nakon registracije. U mogućnosti je da dodaje nove fotografije ili da izabere svoja omiljena lovišta i životinje. Sve podatke korisnik može označiti kao javne, sem imena i prezimena koji su podrazumevano javni.

slika 12: Meni za izbor omiljenih lovišta

slika 11: Forma sa podacima lovca

slika 13: Meni sa prikazom omiljenih lovišta

A picture containing indoor, sitting, person, holding

Description automatically generatedKorisnik može izmeniti svoju šifru klikom na dugme ,,Izmeni“, a potom i na dugme ,,Izmeni lozinku“ koje će ga preusmeriti na novu stranicu. Sva pravila validacije koja su korišćena tokom registracije primenjena su i u ovom slučaju. Takođe, ukoliko korisnik želi da odustane od menjanja lozinke može se vratiti na početnu stranicu klikom na link koji se nalazi u gornjem levom uglu.

slika 14: Forma za pravljenje nove lozinke

### 4.5.3. Podaci korisnika lovačko društvo

Korisnik lovačko društvo nema mogućnosti da registruje svoj nalog, kao što je to slučaj sa korisnikom lovac, već se mora direktno obratiti administratoru. Podaci korisnika lovačko društvo se ne razlikuju preterano od korisnika lovac. Bitna razlika je da broj članova korisnik ne može promeniti, već se on dinamički menja. Broj članova označava vrednost koliko članova lovačkog društva poseduje nalog na portalu.

**A screenshot of a cell phone

Description automatically generated**

slika 15: Forma sa podacima lovačkog društva

### 4.5.4. Galerija slika

A picture containing outdoor, grass, photo, field

Description automatically generatedKlikom na dugme ,,Galerija“, koje se nalazi na dnu forme sa korisničkim podacima, korisnik se preusmerava na stranicu na kojoj može videti svoje slike ili objaviti nove. Galerija je napravljena kao ,,Slide show“ gde se na svakih 5 sekundi korisniku prikazuje sledeća njegova slika. Korisnik može dodavati nove fotografije. Prostor za dodavanje fotografija je ograničen.

slika 16: Galerija slika

### 4.5.5. Vesti i objave

Forma sa vestima prikazuje vesti sortirane po vremenu objavljivanja opadajuće. Naslov i dodatne funkcionalnosti su izmešteni u prvi red panela. Autor objave ima mogućnost da vest arhivira i time onemogući drugim korisnicima da vide vest. Takođe, autor ima mogućnost da zatraži brisanje vesti. Ukoliko niko od drugih korisnika ne uloži prigovor u roku od 24 časa od pristizanja zahteva za brisanje administratoru, administrator sistema može vest obrisati. Takođe, autor može poništiti svoje aktivnosti, ponovo prikazati vest ili poništiti brisanje.

Autor i vreme objavljivanja vesti nalaze se na dnu panela. Kategorija vesti se nalazi na vrhu panela i korisnici mogu izabrati tačno određenu kategoriju vesti koje žele da im se prikažu.

A screenshot of a cell phone

Description automatically generated

slika 17: Panel sa vestima

### 4.5.6. Objavljivanje vesti

A screenshot of a cell phone

Description automatically generatedKlikom na dugme ,,Objavi“, koje se nalazi na vrhu forme vesti, korisniku se pojavljuje forma kao na slici broj 18. Vest se može sastojati od maksimalno 1000 karaktera i može sadržati maksimalno 10 slika. Korisnik objavljuje sve slike odjednom. Jednom pritisnuto dugme ,,Otpremi“ će onemogućiti bilo kakvu promenu. Takođe, korisnik može odrediti vidljivost svoje objave i odabrati kategoriju iz šifrarnika koji uređuje administrator.

slika 18: Forma za objavljivanje nove vesti

Ukoliko korisnik iz padajućeg menija vidljivosti odabere ,,Grupa“, u formi se pojavljuje nov padajući meni. U novom meniju korisnik može odabrati željene korisnike kojima će vest biti vidljiva.

A screenshot of a cell phone

Description automatically generated

slika 19: Biranje grupe korisnika

### 4.5.7. Pretraživanje korisnika

Pretraga korisnika vrši se po njihovom imenu i prezimenu. Može se selektovati koja vrsta korisnika se pretražuje. Ukoliko je neki podatak pretraživanog korisnika privatan, umesto vrednosti podatka, biće prikazana vrednost ,,\*\*\*\*\*\*“. Pretraga korisnika administrator ne postoji, iz razloga što su svi podaci administratora javno dostupni na naslovnoj stranici. S obzirom na to da može postojati više ljudi sa identičnim imenom i prezimenom, ova funkcionalnost pruža korisnicima mogućnost da tačno identifikuju traženog korisnika.

**A screenshot of a cell phone

Description automatically generated**

slika 20: Forma za pretraživanje korisnika

## 4.6. Administratorska stranica

**A screenshot of a cell phone

Description automatically generated** Administrator portala koristi korisničko ime i lozinku kako bi se prijavio na portal. Na početnoj stranici u navigacionom meniju postoji samo jedno dugme, ,,Odjava“ za povratak na naslovnu stranu portala. Slika br, 21 prikazuje podatke administratora koje on ne može direktno na portalu da izmeni. Administrator ima mogućnost da definiše šifrarnik za kategoriju vesti.

slika 21: Forma sa podacima administratora

A screenshot of a cell phone

Description automatically generatedForma na desnoj polovini početne stranice prikazuje vesti koje su označene kao vesti koje su korisnici odabrali za brisanje. Ukoliko niko od drugih korisnika portala nije uložio prigovor na vest koja se briše, administrator može obrisati datu vest. Sa druge strane, ako je neko od korisnika uložio prigovor na brisanje neke vesti, administrator nije u mogućnosti da datu vest obriše. Pored ovih funkcionalnosti, administrator mora komunicirati sa korisnicima putem mejla, kao i dodavati nove korisnike, Lovačka društva.

slika 22: Panel sa vestima za brisanje

## 4.7. Stranica gost

A screenshot of a social media post

Description automatically generatedGosti portala, klikom na dugme ,,Gost“, koje se nalazi u navigacionom baru početne stranice, preusmeravaju se na novu stranicu. Na stranici gost može videte informacije o portalu i podatke o administratorima portala. Pored toga, gost može videti vesti koje su korisnici portala označili kao javne.

slika 23: Stranica gosta

# 5. Realizacija sistema

Ovaj deo rada prikazaće deo realizacije sistema, uz prikaz i objašnjenje delova koda. Takođe, biće prikazani i određeni delovi čija je implementacija stvarala određene probleme.

Web aplikacija je dizajnirana po ugledu na Model View Controller (MVC). Realizacija sistema je jasno podeljena na 3 celine, pri čemu je komponenta za prikaz, View, odvojena i nazivamo je klijentska strana. Model i Controller čine serversku stranu web aplikacije. Korisnik interaguje sa web stranicama, View-om, čiji događaji ,,triggeruju“ odredjene ,,handlere“ koji se nalaze u Contoller celini. Controller komunicira sa bazom podataka i tabele mapira u odgovarajuće klase, Modele.

A picture containing graphical user interface, application

Description automatically generatedServersku stranu aplikacije čine paketi beans, controllers, util i util.dao. U beans paketu nalaze se Model klase koje služe za mapiranje tabela iz baze podataka. Controllers paket sadrži klase koje služe za ,,handlovanje“ događaja, koje korisnik izaziva na klijentskoj strani. U paketu Util.dao su klase koje kontroleri koriste za komunikaciju sa bazom podataka. U paketu util nalaze se klase koje služe za konekciju sa bazom podataka, dohvatanje trenutne HTTP sesije i klasa FilterAutorizacije koja vrši autentikaciju korisnika.

slika 24: Realizacija sistema

## 5.1. Serverska strana aplikacije

### 5.1.1. Modeli

Modeli su klase koje se nalaze u paketu beans i služe za mapiranje tabela iz baze podataka.

### 5.1.2. Kontroleri

Paket kontroleri sadrži klase koje su podeljene tako da svaka klasa odgovara jednoj web stranici. U sledećih nekoliko rečenica biće prikazan kratak opis svih kontrolera.

Jedna od funkcionalnosti koje vrši Administrator kontroler je postavljanje šifrarnika za kategoriju vesti. Kontroler Aktivacija vrši aktivaciju korisnikovog naloga. Nakon registracije, korisniku se šalje verifikacioni link na email adresu. Link u sebi sadrži slučajno generisan ključ, koji kontroler Aktivacija poredi sa ključem koji je sačuvan u bazi. Ovim putem se vrši autentikacija korisnika. Kontoleri Galerija i IzmeniLozinku su pomoćni kontroleri koji služe za prikaz slika korisnika i menjanje korisnikove lozinke. Početna stranica omogućava gostu da vidi određeni sadržaj kao neregistrovani korisnik, način na koji je ovo omogućeno biće opisan u daljem tekstu.

Login je dužan da autentifikuje korisnika. Ukoliko korisnik ima nalog na portalu i on je verifikovan, kontroler Login, na osnovu podataka, korisnika preusmerava na odgovarajuću web stranicu. Registracija kontroler vrši proveru određenih podataka koje korisnik unosi na web stranici sa podacima u bazi, npr. provera da li već postoji isti email u bazi.

A screenshot of a cell phone

Description automatically generated

slika 25: Kontroleri

Kontoleri Lovac i LovackoDrustvo su klase zadužene za gotovo sve aktivnosti korisnika na početnoj stranici portala, napomenućemo samo neke od njih:

* Prikaz kosničkih podataka vrši se pre učitavanja same web stranice. Ova funkcionalnost implementirana je pomoću funkcije init(), uz JSF anotaciju @PostConstruct. Pre samog učitavanja stranice, funkcija iz HTTP sesije dohvata objekat korisnika, koji je zadat kao atribut sesije, i postavlja određene podatke na stranici.
* Prikaz vesti koje korisnik može videti implementiran je takođe pomoću funkcije init(). Lista vesti se dohvata iz baze i u dinamičkom dataScroller-u, koja je komponenta PrimeFaces-a, se prikazuje u posebnoj formi. Kasnije će biti prikazan deo implementacije oviih funkcionalnosti.
* Prikaz rezultata pretraživanja drugih korisnika

### 5.1.3. Paket Util

Util paket sadrži tri klase. Klasa DB vrši konekciju sa bazom podataka. FilterAutorizacije zadužen je za autentikaciju korisnika, dok klasa SessionUtils sadrži funkcije vezane za HTTP sesiju. Jako bitne funkcionalnosti klase SessionUtils su postavljanje i dohvatanje atributa sesije. Sam objekat koji kontroler Login dohvata kao atribut sesije odredjuje na koju stranicu se korisnik preusmerava.

**Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

slika 26: paket Util

slika 26: paket Util

### 5.1.4. Paket Util.dao

Util dao paket sadrži klase putem kojih kotroleri vrše komunikaciju sa bazom podataka. Svaki kontroler poseduje svoju Data access object (DAO) klasu putem koje komunicira sa bazom podataka, dok klasa KorisnickaStranaDAO sadrži metode koje su identične za korisnike lovac i lovačko društvo. KorisnickaStranaDAO sadrži funkcije vezane za dohvatanje podataka iz baze koji se prikazuju na početnoj strani korisnika sistema. Neke od funkcija biće prikazane u nastavku rada.

## 5.2. Klijentska strana

A picture containing graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generatedKlijentska strana aplikacije nalazi se u Web Pages folderu. Sastoji se iz web stranica putem kojih korisnik interaguje sa aplikacijom. Web stranice generišu odgovarajuće događaje koje kontroleri obrađuju. Pored web stranica klijentska strana sadrži sledeće foldere WEB-INF i resources. Folder resources sadrži dva podfoldera, images i css. Folder css sadrži fajlove koje koriste web stranice za pozicioniranje komponenti na stranici, njihovu veličinu, izgled, itd. , dok se u images folderu nalazi slika koja je postavljena kao pozadina na naslovnoj strani. U daljem radu biće dat deo implementacije klijentske strane.

slika 28: Klijentska strana 2

slika 27: Klijentska strana 1

## 5.3. Povezivanje sa bazom

Klasa DB zadužena je za konekciju sa bazom. Poseduje bafer u kome su smeštene konekcije sa bazom, u ovom slučaju maksimalno 50. Ovo znači da maksimalno 50 korisnika može komunicirati sa bazom.

Za ovu web aplikaciju korišćen je JDBC za povezivanje sa bazom. U nastavku prikazan je kod klase DB i primer pristupanja bazi(funkcija za promenu lozinke).

public class DB {

private static DB instance;

private static final int MAX\_CON=5;

private static final Connection[] bafer=new Connection[MAX\_CON];

private int first=0,last=0, free=MAX\_CON;

private DB(){

try{

Class.forName("com.mysql.cj.jdbc.Driver");

for(int i=0; i<MAX\_CON; i++)

bafer[i] = DriverManager.getConnection("jdbc:mysql://localhost:3306/portal?useSSL=false","root","Aleksa-123");

} catch(Exception e){

e.printStackTrace();

}

}

public static DB getInstance(){

if(instance==null) instance=new DB();

return instance;

}

public synchronized Connection getConnection(){

if(free==0) return null;

free--;

Connection con=bafer[first];

first=(first+1)%MAX\_CON;

return con;

}

public synchronized void putConnection(Connection con){

if(con==null) return;

free++;

bafer[last]=con;

last=(last+1)%MAX\_CON;

}

}

Prvo što radimo u funkciji jeste dohvatanje slobodne konekcije iz bafera konekcija. Nakon toga, nad aktivnom konekcijom sa bazom kreiramo objekat tipa PreparedStatement i kao parametar prosleđujemo mu naš SQL upit. Znakovi pitanja označavaju rezervisana mesta za pozicione parametre. Po kreiranju PreparedStatement-a, postavljamo pozicione parametre, ukoliko ih ima, i izvršavamo upit.

public static void izmeniLozinku(String korime, String nova\_lozinka) {

String query1 = "update lovac"

+ " set lozinka = HEX(AES\_ENCRYPT(?,?)) "

+ "where korime = ?";

Connection con = DB.getInstance().getConnection();

PreparedStatement ps1 = null;

try {

ps1 = con.prepareStatement(query1);

ps1.setString(1, nova\_lozinka);

ps1.setString(2, kljuc);

ps1.setString(3, korime);

ps1.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

DB.getInstance().putConnection(con);

if(ps1 != null)

ps1.close();

} catch (SQLException ex) {

Logger.getLogger(LovacDAO.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

## 5.4. Delovi koda implementacije sistema

U narednom delu rada, biće prikazani delovi implementacije sistema. Izabrani delovi u sebi su imali određenu problematiku i bili izazovniji za implementiranje, stoga biće prikazani delovi koda i njihovo objašnjenje.

### 5.4.1. Prijava, registracija i verifikacija

Prilikom prijavljivanja na portal, korisnik unosi svoje korisničko ime i lozinku. Proverom u bazi, utvrđuje se da li korisnik ima nalog i o kom tipu korisnika se radi. Postoje tri tipa korisnika i svaki korisnik ima svoju web stranicu.

Sledeći deo koda prikazuje login funkciju, kontrolera Login. Polja korime i lozinka su polja koja korisnik unosi na klijentskoj strani(View).

public String login() {

Pair<Object, Integer> pair = LoginDAO.dohvatiKorisnika(korime, lozinka);

int tip = 0;

Korisnik k = null;

Drustvo d = null;

Admin a = null;

if (pair != null) {

tip = pair.getValue();

switch(tip){

case 1:

k = (Korisnik) pair.getKey();

break;

case 2:

d = (Drustvo) pair.getKey();

break;

case 3:

a = (Admin) pair.getKey();

break;

}

}

HttpSession sesija = SessionUtils.getSession();

sesija.setAttribute("username", korime);

switch (tip) {

case 0:

alert("Korisnik ne postoji");

return null;

case 1:

sesija.setAttribute("korisnik", k);

return "lovac?faces-redirect=true";

case 2:

sesija.setAttribute("drustvo", d);

return "lovacko\_drustvo?faces-redirect=true";

case 3:

sesija.setAttribute("admin", a);

return "administrator?faces-redirect=true";

case 4:

alert("Nalog nije verifikovan, verifikacioni mail je poslat na vasu email adresu");

return null;

}

return null;

}

Registracija korisnika je složenija, za razliku od login funkcije, potrebno je validirati podatke koje korisnik unosi i proveriti niz podataka i utvrditi njihovu jedinstvenost. Email polje je jedinstveno, kao i polja lovačko društvo i broj kartice lovačkog društva, što znači da ne mogu postojati dva korisnika, koja imaju identične podatke u sistemu. Korisnik mora imati više od 18 godina kako bi mogao da napravio nalog na portalu.

Submit funkcija validira sve podatke koje je korisnik uneo. Ukoliko podaci prodju sve zadate provere, korisniku će na email biti poslat verifikacioni link čije je vreme trajanja 3 dana. Nakon toga, ukoliko korisnik ne verifikuje svoj nalog, on će biti obrisan iz baze. Korisnik se ne može prijaviti na portal bez verifikacije naloga.

public void submit() {

if (!lozinka.equals(ponoviLozinku)) {

alert("Polja lozinke se ne poklapaju");

return;

}

HttpSession sesija = SessionUtils.getSession();

sesija.setAttribute("username", "registracija");

Korisnik k = new Korisnik();

k.setDatumRodjenja(new java.sql.Date(datumRodjenja.getTime()));

k.setDrzava(drzava);

k.setEmail(email);

k.setIme(ime);

k.setImeLovackogDrustva(imeLovackogDrustva);

k.setKorime(korime);

k.setLozinka(lozinka);

k.setMesto(mesto);

k.setPrezime(prezime);

k.setSrednjeIme(srednjeIme);

k.setBrMobilnog(brMobilnog);

k.setBrojClanskeKarte(brojClanskeKarte);

if(status.equals("Neaktivan"))

k.setStatus(false);

else

k.setStatus(true);

k.setKey(getRandomString());

int i = RegistracijaDAO.DodajLovca(k);

switch(i){

case 0:

alert("korisnicko ime vec postoji");

return;

case 1:

alert("email vec postoji");

return;

case 2:

alert("vec postoji korisnik iz istog lovackog drustva sa istim brojem clanske karte");

return;

case 3:

napraviLozinku = false;

pregledPodataka = false;

PosaljiEmail se = new PosaljiEmail(email, k.getKey());

obrisiSve();

alertReplace("Uspesno ste napravili nalog. Verifikacioni email je poslat na Vasu email adresu");

return;

default:

alert("Greska");

return;

}

}

Verifikaciju naloga obavlja kontroler Aktivacija. Link koji je poslat na korisnikovu email adresu sadrži slučajno generisan ključ. Ukoliko se ključ poklapa sa ključem iz baze korisnik će izvršiti uspešnu verifikaciju naloga i moći da se prijavi na sistem putem svog korisničkog imena i lozinke.

@PostConstruct

public void init() {

key = getPassedParameter();

valid = AktivacijaDAO.dohKljuc(key);

invalid = !valid;

}

public String getPassedParameter() {

FacesContext facesContext = FacesContext.getCurrentInstance();

key = (String) facesContext.getExternalContext().

getRequestParameterMap().get("key");

return key;

}

### 5.4.2. Gmail service

Za slanje verifikacionog mejla i mejla vezanog za zaboravljenu lozinku iskorišćen je Google Simple Mail Transfer Protocol(Google SMTP) server. Neophodno je dozvoliti na gmail nalogu posebna podešavanja, kako bi java aplikacija mogla da pristupi nalogu.

public class PosaljiEmail {

final String senderEmailID = "lovacportal.podrska@gmail.com";

final String senderPassword = "Aleksa-123";

final String emailSMTPserver = "smtp.gmail.com";

final String emailServerPort = "587";

String receiverEmailID = null;

static String emailSubject = "Verifikaciona poruka";

String emailBody = "Uspesno ste kreirali nalog. Da biste ga aktivirali kliknite na link:\n http://localhost:8080/projekat/faces/activation.xhtml";

public PosaljiEmail(String receiverEmailID, String key, String message) {

emailSubject = "Zaboravljena lozinka";

// Receiver Email Address

this.receiverEmailID = receiverEmailID;

emailBody = message;

Properties props = new Properties();

props.put("mail.smtp.user", senderEmailID);

props.put("mail.smtp.host", emailSMTPserver);

props.put("mail.smtp.port", emailServerPort);

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.socketFactory.port", emailServerPort);

props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");

props.put("mail.smtp.socketFactory.fallback", "false");

SecurityManager security = System.getSecurityManager();

try {

Authenticator auth = new SMTPAuthenticator();

Session session = Session.getInstance(props, auth);

MimeMessage msg = new MimeMessage(session);

msg.setText(emailBody);

msg.setSubject(emailSubject);

msg.setFrom(new InternetAddress(senderEmailID));

msg.addRecipient(Message.RecipientType.TO,

new InternetAddress(receiverEmailID));

Transport.send(msg);

System.out.println("Message send Successfully:)");

} catch (Exception mex) {

mex.printStackTrace();

}

}

public class SMTPAuthenticator extends javax.mail.Authenticator {

public PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(senderEmailID, senderPassword);

}

}

}

### 5.4.3. Prikazivanje vesti

Vesti se dinamički prikazuju na komponenti Primefaces dataScroller. Komponenta dinamički učitava vesti po deset odjednom. Kada korisnik skroluje formu na dole biće učitano još deset vesti, i tako dok ne budu sve učitane. Ova funkcionalnost komponente dataScroller u velikoj meri ubrzava izvršavanje aplikacije, jer će se stranica učitati iako nisu sve vesti dohvaćene iz baze. Vesti mogu sadržati i do deset slika. Slike se prikazuju pomoću komponente ,galleria, koja podržava AJAX. Ukoliko korisnik skroluje do vesti koja sadrži slike, slike se prikazuju u vidu slideshow-a ili ih korisnik može menjati manuelno.

Korisnik može da vidi sve vesti koje su javne ili njemu namenjene. Ukoliko je neka od vesti, vest koju je on objavio tada ima mogućnost da vest arhivira ili zatraži njeno brisanje. Arhiviranu vest može videti samo korisnik koji ju je objavio, dok je brisanje vesti moguće samo ukoliko se niko od ostalih korisnika ne bude žalio i ukoliko sam administrator odluči da odobri zahtev.

Biće prikazan deo koda koji za odgovarajući tip korisnika, i odgovarajući id korisnika dohvata vesti iz baze. Takođe, biće prikazana i klijentska strana, dataScroller komponenta.

public static List<Vest> dohVesti(int id, boolean tip\_korisnika, int kategorija){

String query1 = "select \* " //dohvatam vesti dostupne svim lovcima

+ "from vesti "

+ "where privatnost in (?,4) and (arhivirana < 1 or (arhivirana > 0 and tip\_korisnika = ? and id\_korisnika = ?)) "

+ "order by datum desc";

String query2 = "select DISTINCT(v.id), v.naziv, v.tekst, v.kategorija, v.datum, v.autor, v.id\_korisnika, v.arhivirana, v.zahtev\_za\_brisanje, v.flag\_obrisi " //dohvatam vesti koje su dostupne ovom korisniku jer je to neko selektovao

+ "from vesti v, nivo\_vesti nv "

+ "where v.privatnost = 3 and ((v.arhivirana < 1 and nv.tip\_korisnika = ? and nv.id\_korisnika = ? and nv.id\_vesti = v.id) or (v.tip\_korisnika = ? and v.id\_korisnika = ?)) ";

//(arhivirana < 1 or (arhivirana > 0 and tip\_korisnika = 0 and id\_korisnika = ?)) ovo proveravam da li je arhivirana i ako jeste

//onda samo korisnik moze da vidi ovu vest

String query3 = "select slika "

+ "from slike\_vest "

+ "where id\_vesti = ?";

ArrayList<Vest> vesti = new ArrayList<>();

Connection con = DB.getInstance().getConnection();

PreparedStatement ps1 = null;

PreparedStatement ps2 = null;

PreparedStatement ps3 = null;

ResultSet rs = null;

ResultSet rs3 = null;

try {

ps1 = con.prepareStatement(query1);

ps2 = con.prepareStatement(query2);

ps3 = con.prepareStatement(query3);

if(tip\_korisnika)

ps1.setInt(1, 2);

else

ps1.setInt(1, 1);

ps1.setBoolean(2, tip\_korisnika);

ps1.setInt(3, id);

ps1.execute();

rs = ps1.getResultSet();

while(rs.next()){

Vest v = new Vest();

v.setAutor(rs.getString("autor"));

v.setDatum(rs.getTimestamp("datum"));

v.setNaziv(rs.getString("naziv"));

v.setTekst(rs.getString("tekst"));

v.setKategorija(rs.getInt("kategorija"));

v.setId(rs.getInt("id"));

v.setId\_korisnika(rs.getInt("id\_korisnika"));

v.setArhivirana(rs.getBoolean("arhivirana"));

v.setZahtev\_za\_brisanje(rs.getBoolean("zahtev\_za\_brisanje"));

v.setFlag\_obrisi(rs.getBoolean("flag\_obrisi"));

//dohvatam iz baze slike ako ih ima u vesti

ps3.setInt(1, v.getId());

ps3.execute();

rs3 = ps3.getResultSet();

List<byte[]> slike = new ArrayList<>();

while(rs3.next()){

slike.add(rs3.getBlob("slika").getBytes(1, (int) rs3.getBlob("slika").length()));

}

v.setSlike(slike);

if(kategorija == 0 || kategorija == v.getKategorija())

vesti.add(v);

else

if(kategorija == 3 && v.isZahtev\_za\_brisanje())

vesti.add(v);

}

ps2.setBoolean(1, tip\_korisnika);

ps2.setInt(2, id);

ps2.setBoolean(3, tip\_korisnika);

ps2.setInt(4, id);

ps2.execute();

rs = ps2.getResultSet();

while(rs.next()){

Vest v = new Vest();

v.setAutor(rs.getString("autor"));

v.setDatum(rs.getTimestamp("datum"));

v.setNaziv(rs.getString("naziv"));

v.setTekst(rs.getString("tekst"));

v.setKategorija(rs.getInt("kategorija"));

v.setId(rs.getInt("id"));

v.setId\_korisnika(rs.getInt("id\_korisnika"));

v.setArhivirana(rs.getBoolean("arhivirana"));

v.setZahtev\_za\_brisanje(rs.getBoolean("zahtev\_za\_brisanje"));

v.setFlag\_obrisi(rs.getBoolean("flag\_obrisi"));

//dohvatam iz baze slike ako ih ima u vesti

ps3.setInt(1, v.getId());

ps3.execute();

rs3 = ps3.getResultSet();

List<byte[]> slike = new ArrayList<>();

while(rs3.next()){

slike.add(rs3.getBlob("slika").getBytes(1, (int) rs3.getBlob("slika").length()));

}

if(slike != null)

v.setSlike(slike);

if(kategorija == 0 || kategorija == v.getKategorija())

vesti.add(v);

else

if(kategorija == 3 && v.isZahtev\_za\_brisanje())

vesti.add(v);

}

Collections.sort(vesti, new Comparator<Vest>() {

@Override

public int compare(Vest v1, Vest v2){

return v2.getDatum().compareTo(v1.getDatum());

}

});

return vesti;

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

DB.getInstance().putConnection(con);

if (ps2 != null) {

ps2.close();

}

if (ps1 != null) {

ps1.close();

}

if (rs != null) {

rs.close();

}

} catch (SQLException ex) {

Logger.getLogger(RegistracijaDAO.class.getName()).log(Level.SEVERE, null, ex);

}

}

return vesti;

}

Klijentska strana:

<p:dataScroller id="vesti" value="#{lovac.vesti}" var="vest" chunkSize="10" rendered="#{!lovac.objavi and !lovac.pretraga}">

<f:facet name="header">

<p:commandButton value="Objavi" action="#{lovac.objaviVest()}" class="button-vesti" update="novosti"/>

<p:outputLabel value="Vesti" style="padding-right: 20%; "/>

<p:outputPanel style="display: inline-block; float: left;">

<p:selectOneMenu value="#{lovac.kategorija}" >

<f:selectItem itemLabel="Sve vesti" itemValue="" />

<f:selectItem itemLabel="Oznacene za brisanje" itemValue="brisanje" />

<f:selectItems value="#{lovac.sve\_kategorije\_vesti}" />

<p:ajax event="change" update="vesti" listener="#{lovac.updateVesti()}" />

</p:selectOneMenu>

</p:outputPanel>

<br/>

</f:facet>

<p:panel id ="panel\_vest" widgetVar="#{vest.id}" toggleable="true" style="display: inline-block; width: 100%;">

<f:facet name="header">

<p:outputLabel value="#{vest.naziv}" />

<p:outputLabel value="(Vest je arhivirana)" rendered="#{vest.arhivirana}" />

<p:outputLabel value="(Vest ce biti obrisana)" rendered="#{vest.zahtev\_za\_brisanje}" />

</f:facet>

<f:facet name="options" >

<p:menu>

<p:menuitem value="Obrisi" action="#{lovac.zahtevZaBrisanje(vest.id)}" rendered="#{lovac.proveriAutora(vest.id\_korisnika) and !vest.zahtev\_za\_brisanje}" update="vesti"/>

<p:menuitem value="Ponisti brisanje" action="#{lovac.ponistiBrisanje(vest.id)}" rendered="#{lovac.proveriAutora(vest.id\_korisnika) and vest.zahtev\_za\_brisanje}" update="vesti" />

<p:menuitem value="Arhiviraj" action="#{lovac.arhiviraj(vest.id)}" rendered="#{lovac.proveriAutora(vest.id\_korisnika) and !vest.arhivirana}" update="vesti"/>

<p:menuitem value="Prikazi" action="#{lovac.prikazi(vest.id)}" rendered="#{lovac.proveriAutora(vest.id\_korisnika) and vest.arhivirana}" update="vesti"/>

<p:menuitem value="Prigovor" action="#{lovac.prigovorNaBrisanje(vest.id)}" rendered="#{!lovac.proveriAutora(vest.id\_korisnika) and vest.zahtev\_za\_brisanje}" update="vesti"/>

</p:menu>

</f:facet>

<p:outputLabel value="\*#{lovac.dohvatiKategorijuVesti(vest.kategorija)}\*" style="text- decoration:underline; font-weight: bold;"/>

<br /><br />

<h:outputText value="#{vest.tekst}" escape="false" />

<p:galleria value="#{vest.slike}" var="slika" showCaption="true" rendered="#{not empty vest.slike}">

<oou:graphicImage class="img" value="#{slika}" dataURI="true" title="Slika"/>

</p:galleria>

<br/><br/>

<p:outputLabel value="#{lovac.dohvatiAutora(vest.id)}" style="display: inline-block; float: left; text- align: right;"/>

<p:outputLabel value="#{vest.datum}" style="display: inline-block; float: right; text-align: left;">

<f:convertDateTime pattern="hh:mm a dd/MM/yyyy" />

</p:outputLabel>

</p:panel>

<br/><br/>

</p:dataScroller>

### 5.4.4. Pretraživanje korisnika

Prijavljeni korisnik ima mogućnost pretrage drugih korisnika putem njihovog imena, za lovačka društva, ili imena i prezimena, za korisnika lovac. Korisnik može sam izabrati koje podatke želi da deli sa drugim korisnicima, a koje ne. Ime i prezime su podrazumevano deljeni podaci. Prikazaćemo funkciju koja dohvata tražena lovačka društva čije se ime sastoji iz ključne reči koju je korisnik uneo. Vidljivost podataka implementirana je pomoću celog, pozitivnog broja. Svaki bit broja u binarnoj predstavi određuje da li je neki podatak vidljiv ili ne.

public static List<Drustvo> pretraziLovackaDrustva(String kljucnaRec) {

String query1 = "select \* "

+ "from lovacko\_drustvo "

+ "where ime like ";

Connection con = DB.getInstance().getConnection();

PreparedStatement ps1 = null;

ResultSet rs = null;

ArrayList<Drustvo> lista = new ArrayList<>();

try {

query1 += "'" + kljucnaRec + "%'";

ps1 = con.prepareStatement(query1);

ps1.execute();

rs = ps1.getResultSet();

if(rs == null)

return null;

while(rs.next()){

Drustvo d = new Drustvo();

d.setIme(rs.getString("ime"));

d.setEmail(rs.getString("email"));

d.setStatus(rs.getBoolean("status"));

d.setBroj\_clanova(rs.getInt("broj\_clanova"));

d.setIznos\_clanarine(rs.getInt("iznos\_clanarine"));

Pair<MenuModel, List<String>> pair2 = KorisnickaStranaDAO.dohOmiljeneZivotinje(rs.getInt("id"), true);

d.setOz\_model(pair2.getKey());

int val = rs.getInt("vidljivost\_podataka");

if((val>>1 & 0x1) == 1)

d.setPrikaziEmail(true);

if((val>>2 & 0x1) == 1)

d.setPrikaziStatus(true);

if((val>>3 & 0x1) == 1)

d.setPrikaziBrojClanova(true);

if((val>>4 & 0x1) == 1)

d.setPrikaziIznosClanarine(true);

if((val>>5 & 0x1) == 1)

d.setPrikaziOZ(true);

lista.add(d);

}

return lista;

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

DB.getInstance().putConnection(con);

if (ps1 != null) {

ps1.close();

}

} catch (SQLException ex) {

Logger.getLogger(RegistracijaDAO.class.getName()).log(Level.SEVERE, null, ex);

}

}

return null;

}

# 6. Zaključak

U radu je prikazana izrada jedne moderne web aplikacije. Opisane su korišćene tehnologije i alati, među kojima dominira korišćenje JSF framework-a. Najpre su prikazani zahtevi sistema, zahtevi korisnika lovac i lovačko društvo, i zahtevi modula administrator. Nakon toga dat je opis tehnologija i alata korišćenih pri izradi aplikacije, gde su prikazane njihove prednosti i mane. Korišćenje JSF framework-a, uz primese PrimeFaces-a i OmniFaces-a, dosta je jednostavnije od pisanja čistog HTML-a. PrimeFaces pruža UI koji je vrlo lako nadograditi, uz CSS, i dobiti veoma dobar izgled frontend aplikacije. Ovo ostavlja prostora programeru da se fokusira na backend deo aplikacije. Primefaces komponente korišćene su uz primenu AJAXA, za koji je detaljno objašnjeno koje funkcionalnosti pruža u vidu pravljenja što dinamičnije aplikacije. Uz AJAX dogadjaje, na frontendu, upotrevljen je JavaScript i time dosta smanjena komunikacija serverske i klijentske strane, što dovodi do značajnog smanjenja odziva aplikacije. Dinamički prikaz slika u aplikaciji omogućen je primenom galerije PrimeFaces-a i graphicImage taga iz biblioteke OmniFaces.

Backend deo aplikacije implementiran je uz korisćenje Java programskog jezika, dok je za komuniciranje sa bazom korišćen Java Database Connectivity(JDBC). Prednost JDBC-a, za razliku od nekog object-relational mapping modela (ORM), je u tome što se koristi čisti SQL. Programer jasno vidi napisan upit i vrlo lako može uočiti greške, ukoliko ih ima. Takođe, velika prednost je mogućnost naknadnog prosleđivanja parametara. Izvršavanjem upita, korišćenjem objekta PreparedStatement, prilikom dohvatanja podataka iz tabele, podaci se smeštaju u posebne klase koje nazivamo Java bean-ovi. Kontroleri, uz JSF anotacije @MenagedBean i @Named, omogućavaju lak pristup svojim poljima i funkcijama na klijentskoj strani. Uz ove dve anotacije, često se koriste anotacije za opseg dostupnosti, kao što su @SessionScoped, @RequestScope, @ViewScope, itd. Izdvojio bih onu koja je korišćena u implementaciji kontrolera Lovac, a to je @ViewScope anotacija. Korišćenjem anotacije @PostConstruct pri svakom osvežavanju stranice ili neke forme na stranici poziva se init() funkcija. Ova funkcionalnost je primenjena kod objavljivanja vesti. Kada korisnik objavi vest, vest se ubacuje u bazu podataka. Da bi vest bila vidljiva korisniku tokom date sesije, neophodno je ponovo dohvatiti sve vesti iz baze i prikazati ih. Pri objavljivanju vesti, refrešujemo stranicu primenom JavaScript funkcije, nakon koje se funkcija init() automatski poziva i dohvata listu vesti iz baze.

Softverski sistem za povezivanje lovaca je portal namenjen prikupljanju i razmeni podataka o bivšim i sadašnjim lovcima i lovačkim društvima, kao i uspostavljanju njihovih međusobnih veza. Korisnici aplikacije, lovac i lovačko društvo, mogu pretraživati korisnike, menjati privatne podatke, objavljivati vesti i fotografije. Funkcionalnost objavljivanja vesti podržava mogućnost pravljenja grupe ljudi kojima će vest biti vidljiva. Pored svih funkcionalnosti koja ova web aplikacija pruža, uvek postoji prostor za nadogradnju. Jedna od njih bila bi implementacija funkcionalnosti za razmenu poruka između korisnika. Ovaj vid funkcionalnosti može lako biti implementiran korišćenjem PrimeFaces Push-a koji koristi Atmosphere framework. Na ovaj način bi aplikacija koja već dosta nudi bila podignuta na viši nivo.

# 7. Literatura

[1]<https://runestone.academy/runestone/books/published/webfundamentals/WWW/history.html#:~:text=Tim%20Berners-Lee%20invented%20the,physics%20laboratory%20near%20Geneva%2C%20Switzerland>.

[2] <http://www.observationalhazard.com/2018/06/history-of-web-programming.html>

[3]<https://en.wikipedia.org/wiki/Ajax_(programming)#:~:text=In%201998%2C%20the%20Microsoft%20Outlook,Explorer%205.0%20in%20March%201999.&text=This%20increased%20interest%20in%20AJAX%20among%20web%20program%20developers>.

[4] <https://stackoverflow.com/questions/8694922/whats-the-difference-between-a-web-site-and-a-web-application>

[5] <https://dinarys.com/blog/website-vs.-webapplication>

[6] <https://en.wikipedia.org/wiki/NetBeans>

[7]<https://en.wikipedia.org/wiki/Java_Database_Connectivity#:~:text=Java%20Database%20Connectivity%20(JDBC)%20is,Edition%20platform%2C%20from%20Oracle%20Corporation>.

[8]<https://www.mysql.com/products/workbench/#:~:text=MySQL%20Workbench%20is%20a%20unified,Linux%20and%20Mac%20OS%20X>.

[9] <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard>

[10] <https://en.wikipedia.org/wiki/Jakarta_Server_Faces>

[11] <http://www.javaserverfaces.org/>

[12] <https://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html>

[13] <https://www.primefaces.org/documentation/>

[14] <https://showcase.omnifaces.org/>

[15] https://www.oracle.com/java/technologies/java-message-service.html