Dokumentacija

Ova web aplikacija je namijenjena da pomogne pri vođenju neke firme. Pomoću ove aplikacije korisnik može lakše da prati evidenciju zaposlenih. Ime, prezime, godine, E-mail, grana poslovanja, ID zaposlenog, plata, spol, pozicija i broj telefona su informacije koje korisnik može da unese. Sve unseen informacije se prikazuju, također se iste mogu obrisati ili dodati. Korisnik može da bude samo neko ko ima pristup šifri i korisničkom imenu koji se nalaze u bazi podataka. Korisnik ima i pregled grana poslovanja i koja grana poslovanja posjeduje koji ID. Također ima, izdvojen pregled, E-mail kontakt svakog zaposlenog. Naravno u svakom trenutku korisnik može da se odjavi iz aplikacije.

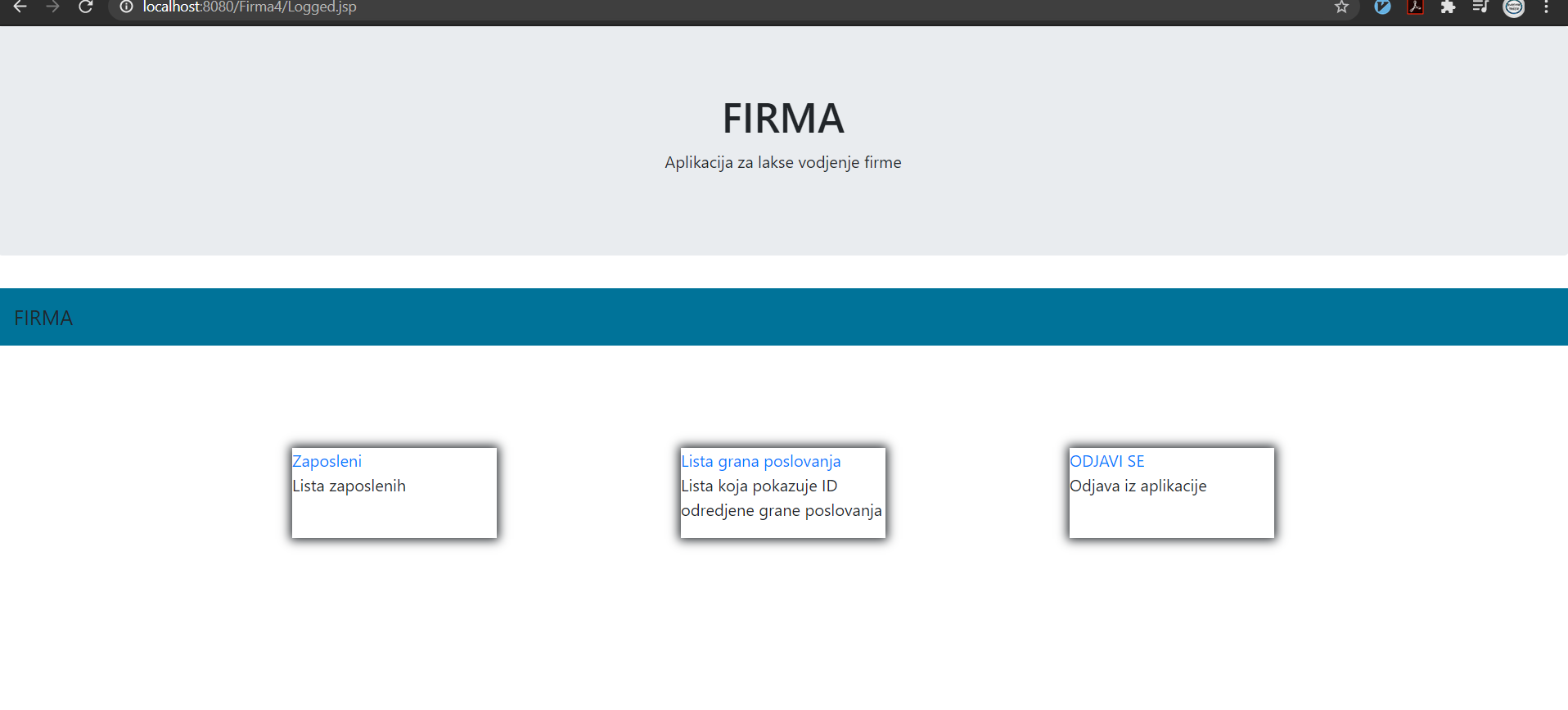
1. **Uputstvo za korišćenje aplikacije od strane korisnika:**

Pristupiti aplikaciji mogu samo osobe koje dobiju ovlašćenje tj. korisničko ime i šifru.

Korisničko ime: **amar** i Šifra: **sifra**

Korisničko ime: **neko** i Šifra: **sifra**

će dopustiti dalji pristup aplikaciji, gdje korisnik nailazi na „glavnu“ stranicu. Koja izgleda ovako:



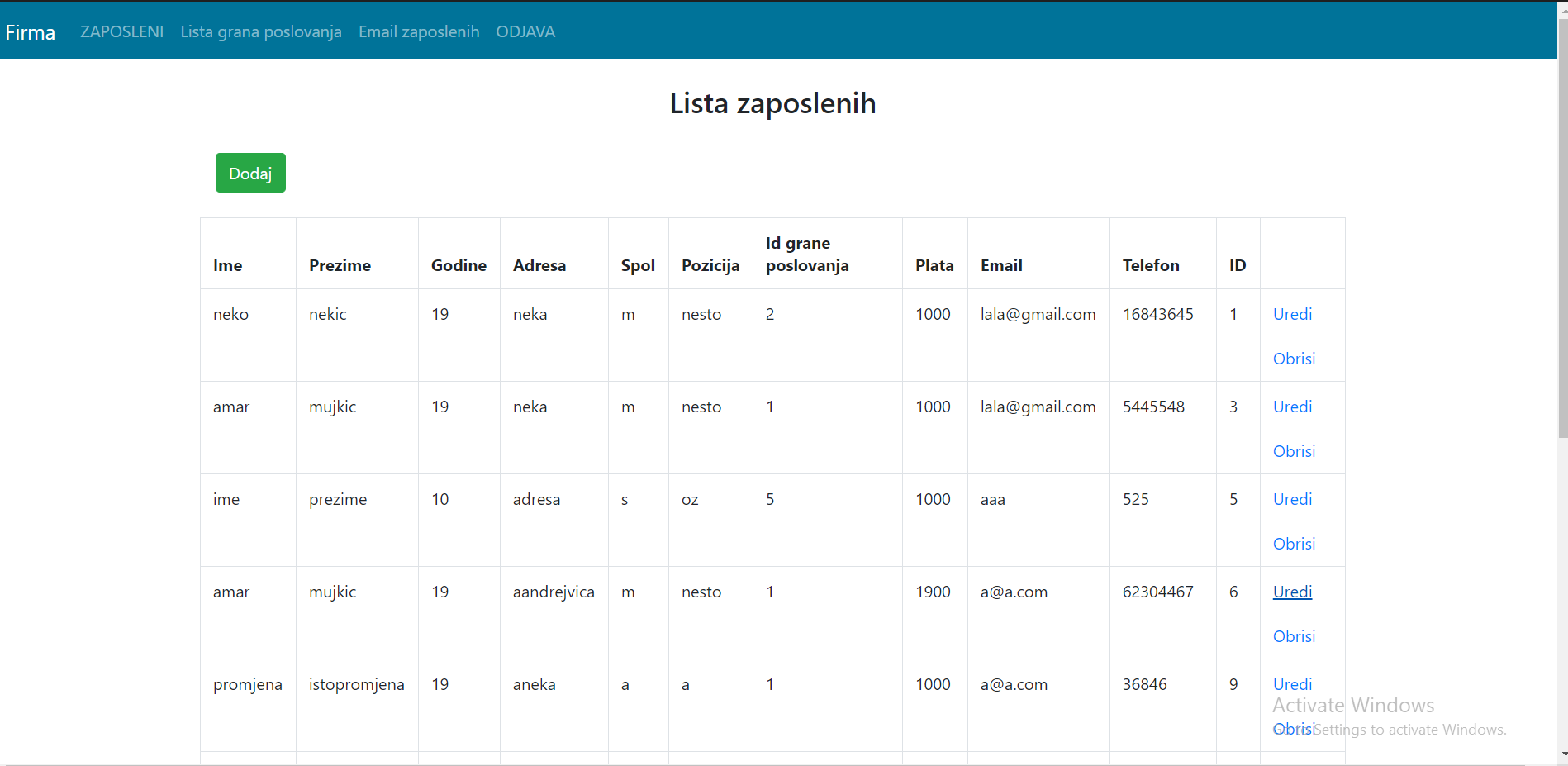
na njoj se nalaze tri kvadratića, u svakom od njih nalazi se objašnjenje za njihove funkcije.

Prvi kvadratić daje upute da klikom na „**zaposleni**“ dolazimo do stranice koja ustvari sadrži glavnu listu u programu tj. listu zaposlenih.

Drugi kvadratić „**Lista grana poslovanja**“ vodi na listu grana poslovanja.

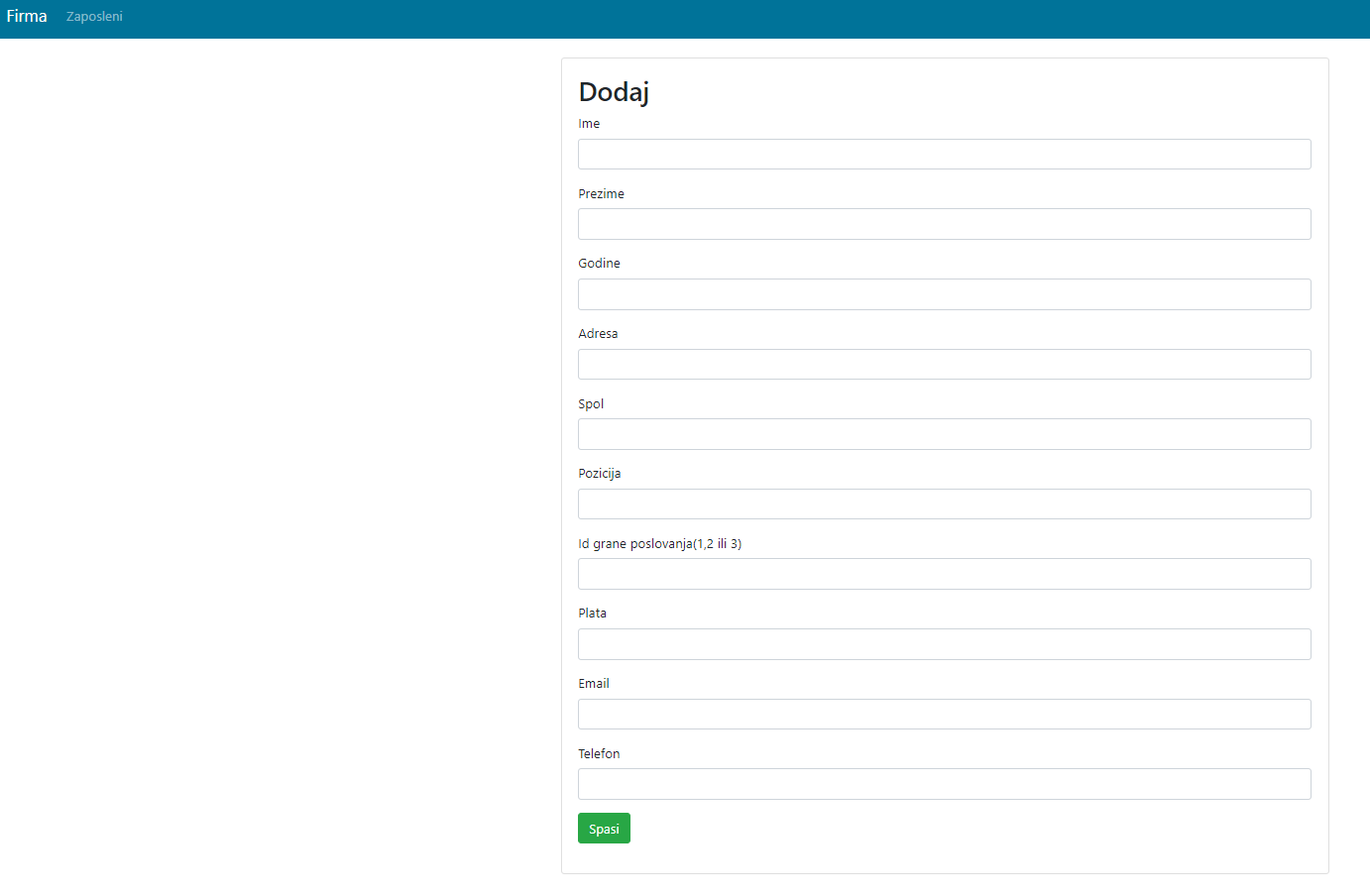
Treći kvadratić „**ODJAVI SE**“ odjavljuje korisnika iz aplikacije i traži ponovni upis Korisničkog imena i šifre koja mu odgovara.

Kada smo kliknuli na „zaposleni“ nailazimo na ovakvu stiuaciju:

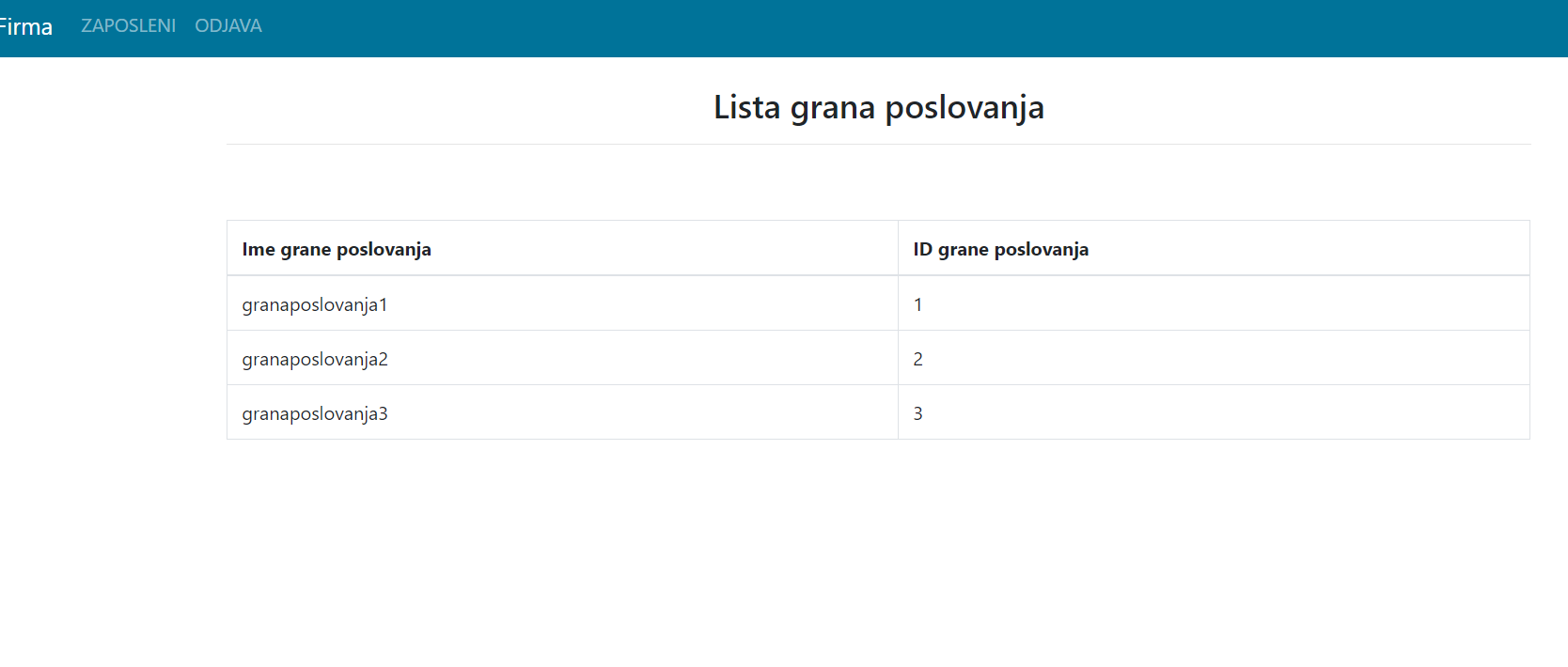


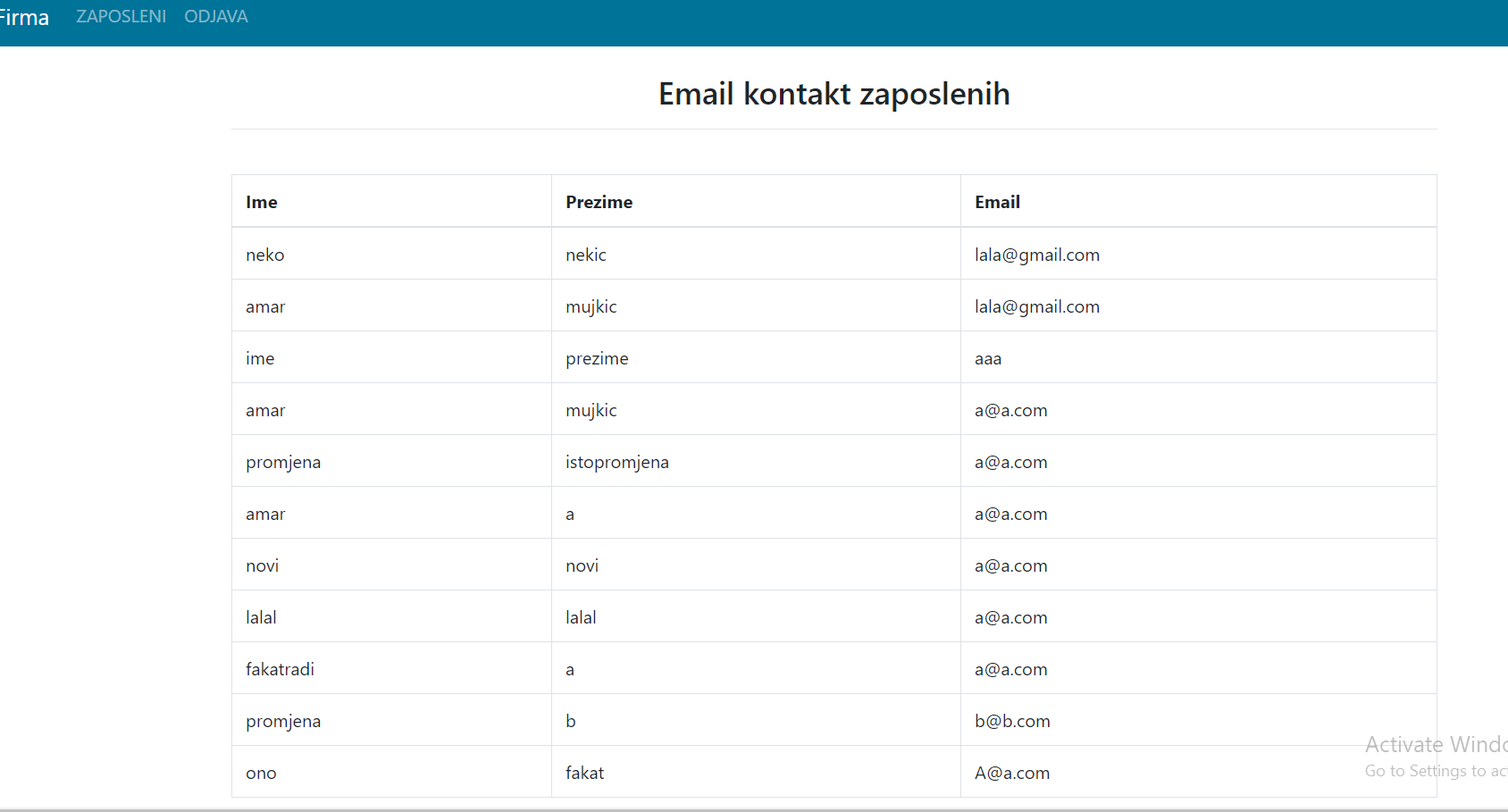
Gledajući s vrha u plavoj „traci“ nalazi se 5 funkcija. Klikom na **Firma** vraćamo se na „početnu“ stranicu. Klikom na „**Lista grana poslovanja**“ prelazimo na stranicu, na kojoj se nalazi tabela imena i ID grana poslovanja. S obzirom na to da u tabeli gdje imamo sve informacije o nekom zaposlenom se nalazi samo ID grane poslovanja, postoji tabela koja govori šta čemu pripada. Opcija „**Email zaposlenih**“ nas vodi na stranicu koja sadrži tabelu sa svim imenima i prezimenima zaposlenih kao i njihovog E-maila. I „**ODJAVA**“ naravno vraća na „login“ tj. traži se ponovni unos podataka za pristup.

Ispod natpisa „Lista zaposlenih“ nalazi se zeleno dugme „**Dodaj**“ koje nas vodi na stranicu koja će da nam dopusti unos podataka nekog novog zaposlenog. Sličnu funkciju imamo i u tabeli klikom na „**Uredi**“ dolazimo na istu stranicu tj. onu gdje unosimo podatke, samo ovom prilikom određeni podaci će već biti ispunjeni što znaći da Uredi funkcija, služi za ispravljanje već unesenih informacija. Funkcijom „**Obriši**“ brišemo sve podatke o jednom zaposlenom.



Ovo je stranica u kojoj popunjavamo informacije o korisniku. Sve osim ID zaposlenog, unosi korisnik. ID zaposlenog se generiše sam i svaki zaposleni ima svoj zaseban ID. Na ovoj stranici postoje dvije opcije u plavoj „traci“. To su „Firma“ i „zaposleni“ , firma nas vraća na početnu stranicu dok zaposleni vraća na glavnu tabelu tj. listu zaposlenih. Klikom na dugme „**Spasi**“ podaci o korisniku se spašavaju i prikazuju u tabeli.

Lista grana poslovanja izgleda ovako: 

U ovu tabelu korisnik ne upisuje ništa. U ovoj tabeli korisnik ne može da pravi bilo kakve promjene. U plavoj traci opet ima određene opcije, koje smo već pomenuli. Povratak na početnu stranicu, povratak na tabelu „Lista zaposlenih“ i odjava iz aplikacije. 

Ova slika predstavlja tabelu u koj se nalaze E-mail kontakti zajedno sa imenima i prezimenima svih zaposlenih. Takođe ima iste funkcije kao i prethodna stranica i korisnik ne može da mijenja podatke unutar tabele.

1. **KOD**

Program se sastoji od:

7 jsp file-ova,

4 paketa u sourceu u kojima se nalazi 6 java klasa.

U klasi „**User**“ se nalaze varijable i konsrtuktori zajedno sa geterima i seterima. Dok se u „UserDAO“ nalazi većina logike potrebne za rad programa. Ova DAO klasa sadrži varijable potrebne za konekciju baze podataka iz koje i u koju upisujemo/ispisujemo podatke. Nakon konekcije nalaze se „static sql statements“ koji služe za upravljanje bazom podataka.

**private static final String INSERT\_USERS\_SQL = "INSERT INTO zaposleni" +**

**"(Firstname, Lastname, Age, Adress, Sex, post, branchId,Salary,Email,Phone) VALUES" + "(?,?,?,?,?,?,?,?,?,?);";**

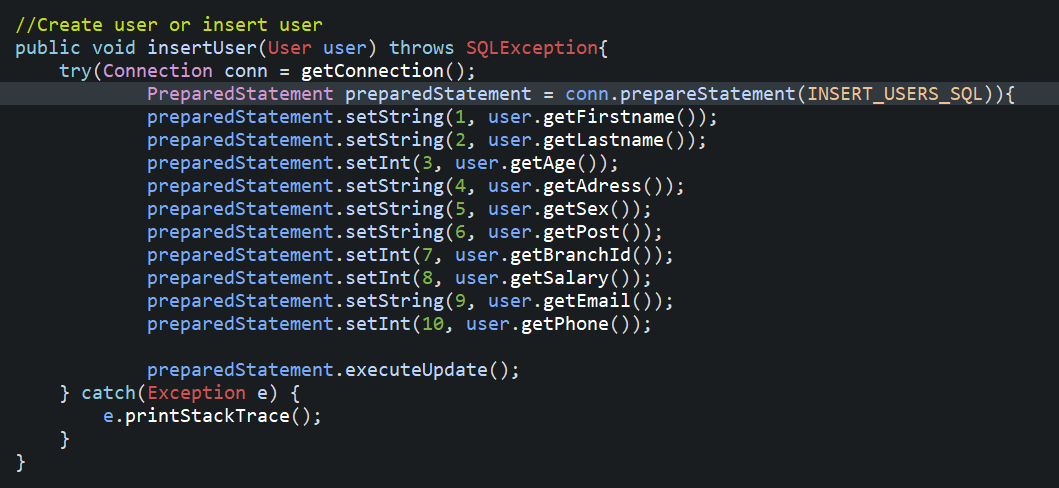
**private static final String BRANCH= "select \* from gp";**

**private static final String SELECT\_USER\_BY\_ID = "select Firstname, Lastname, Age, Adress, Sex, post, branchId,Salary,Email,Phone from zaposleni where EmpId=?";**

**private static final String SELECT\_ALL\_USERS = "select \* from zaposleni";**

**private static final String DELETE\_USERES\_SQL = "delete from zaposleni where Empid =?;";**

**private static final String UPDATE\_USERS\_SQL = "update zaposleni set Firstname=?, Lastname=?, Age=?, Adress=?, Sex=?, post=?, branchId=?, Salary=?, Email=?, Phone=? where EmpId =?;";**

Znači svaka od njih upravlja bazom podataka, npr. INSERT\_USERS\_SQL unosi određene vrijednosti u bazu podataka, koje dobijemo prilikom unosa korisnika. BRANCH pokazuje imena i ID određene grane poslovanja. SELECT\_USER\_BY\_ID služi za odabir samo onih podataka kod kojih se ID podudara sa traženim ID. SELECT\_ALL\_USERS pokazuje listu svih informacija iz tabele “zaposleni“. DELETE\_USERS\_SQL briše podatke koji se podudaraju sa određenim ID. UPDATE\_USERS\_SQL unosi nove podatke na mjesto podataka koji se podudaraju sa određenim ID. U daljem kodu se nalazi metoda (getConnection) za uspostavljanje veze sa bazom podataka, koju ćemo da koristimo u naredim metodama. 

U metodi insertUser dajemo user objekat kao parametar. Sa „try“ automatski zatvaramo konekciju i pozivamo konekciju. Dalje nam PreparedStatement pomaže da se izvrši sql query tj da se izvrši INSERT\_USERS\_SQL koji će da unese podatke u bazu podataka. preparedStatement.executeUpdate(); izvršava sve ovo. Sa metodom catch hvatamo greske i ispisujemo. Sve metode funkcionišu na slične načine.

Update metoda funkcioniše na isti način, samo umjesto catch-a vraćamo rowUpdated tj. boolean jer je sama update metoda boolean. Npr. u selectUser imamo array listu koja nam pomaže pri ispisu informacija u određenom jsp fajlu u kojem se nalazi odgovarajuća tabela. Branches također uz konekciju i array listu ispisuje u jsp-u. DeleteUser je boolean kao i update.

UserServlet je klasa koja se koristi za proširenje mogućnosti servera koji hostuju aplikacije kojima se pristupa pomoću programskog modela request-response. Sve requeste „handel-amo“ kroz jednu doPost metodu, „call-amo“ tj. pozivamo metodu this.doGet(request,response). U doGet smo napravili switch statement za akcije koji “handel-a” sve zahtjeve(request-e).



Action dobijamo kroz request objekat (String action=request.getServletPath();) korišćenjem “getServletPath”. U metodi showNewForm postoji dispatcher koji Definira objekt koji prima zahtjeve od klijenta i šalje ih bilo kojem resursu (kao što je servlet, HTML datoteka ili JSP datoteka) na poslužitelju. Što znači da kada god pozovemo url ”/new” proslijedit će user form objekat u jsp stranicu.

private void showNewForm(HttpServletRequest request, HttpServletResponse response)

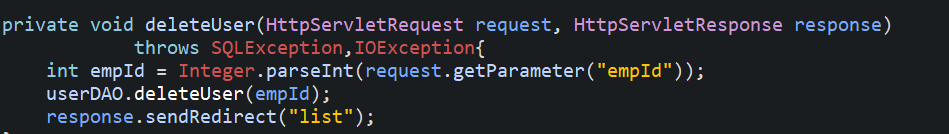
throws ServletException, IOException{

RequestDispatcher dispatcher = request.getRequestDispatcher("UserForm.jsp");

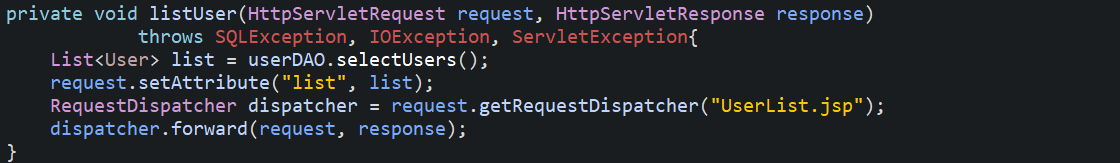
dispatcher.forward(request, response);

}



U insertUser metodi izdvajamo podatke zaposlenog tj. user objekta i prosljeđujemo konstruktoru. I pravimo novi user. Na kraju pozivamo insertUser metodu od userDAO objekta da bi spasili određenog zaposlenog tj. objekat user u bazu podataka. I redirect-amo, odnsono prosljeđujemo listu. 

deleteUser metoda briše user-a, tj zaposlenog sa odgovarajućim ID iz baze podataka. I redirect-a na listu.



U listUser metodi hvatamo sve user objekte i sve ih stavljamo na listu “list” uz pomoć setAttribute metodu. I onda dispatcher prosljeđuje na jsp stranicu.

private void updateUser(HttpServletRequest request, HttpServletResponse response)

throws SQLException,IOException{

int EmpId = Integer.parseInt(request.getParameter("empId"));

String Firstname = request.getParameter("Firstname");

String Lastname= request.getParameter("Lastname");

String Email = request.getParameter("Email");

String post = request.getParameter("post");

int Age = Integer.parseInt(request.getParameter("age"));

String Adress = request.getParameter("Adress");

String Sex = request.getParameter("Sex");

int branchId = Integer.parseInt(request.getParameter("branchId"));

int Salary = Integer.parseInt(request.getParameter("salary"));

int Phone = Integer.parseInt(request.getParameter("phone"));

User user = new User(Firstname, Lastname, Age , Adress, Sex, post, branchId, Salary, Email, Phone,EmpId);

userDAO.updateUser(user);

response.sendRedirect("list");

}

updateUser funkcioniše slično kao i inserUser, uz pomoć request objekta i getparametar metode pozivamo podatk. Opet pravimo user objekat. Update-uje samo podatke onog zaposlenog koji ima odgovarajući ID. Sve metode funkcionišu na sličan način.

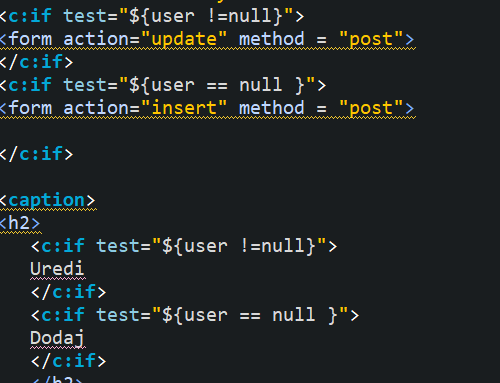
Sve metode pozivamo u switchu. Default case u kosturu switch-a će da proslijedi listu tj user list stranicu na kojoj se nalaze svi podaci u tabeli.

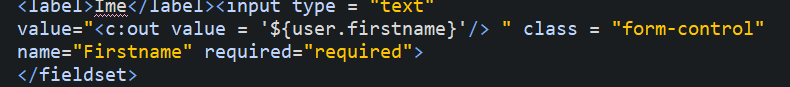
**Error.jsp**

Ako se desi neki “exception” prikazuje se na error.jsp stranici.

**UserForm.jsp**

UserForm stranica prikazuje input. I traži sve potrebne informacije koje će se unijeti u bazu podataka.



UserForm ima dvije funkcionalnosti. Ako user nije null onda se ponaša kao update stranica koja uređuje već unesene podatke i dešava se akcija “update”. Ako je user null onda se ponaša kao input tj.stranica koja traži nove podatke i pravi novog user-a tj. zaposlenog i dešava se akcija “insert”. Korišćeni su bootstrapsi CSS-a. svaki input je fieldset.

Na kraju stranice imamo jedno dugme koje će spasi formu.

**UserList.jsp**

Ova jsp stranica kao i sve ostale koristi isti bootstrap sa cdn linkom. Na ovoj stranici se nalazi tabela sa stupcima i redovima

.

<table class="table table-bordered">

<thead>

<tr>

<th>Ime </th>

<th>Prezime </th>

<th>Godine </th>

<th>Adresa </th>

<th>Spol </th>

<th>Pozicija </th>

<th>Id grane poslovanja </th>

<th>Plata </th>

<th>Email </th>

<th>Telefon </th>

<th>ID</th>

<th></th>

</tr>

</thead>

<tbody>

<**c:forEach** var="user" items="${list}">

<tr>

<td><**c:out** value="${user.firstname}" /></td>

<td><**c:out** value="${user.lastname}" /></td>

<td><**c:out** value="${user.age}" /></td>

<td><**c:out** value="${user.adress}" /></td>

<td><**c:out** value="${user.sex}" /></td>

<td><**c:out** value="${user.post}" /></td>

<td><**c:out** value="${user.branchId}" /></td>

<td><**c:out** value="${user.salary}" /></td>

<td><**c:out** value="${user.email}" /></td>

<td><**c:out** value="${user.phone}" /></td>

<td><**c:out** value="${user.empId}" /></td>

<td><a href="edit?empId=<c:out value='${user.empId}' />">Uredi</a>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <a

href="delete?empId=<c:out value='${user.empId}' />">Obrisi</a></td>

</tr>

## </c:forEach>

Prosljeđujemo određene vrijednosti u svaki red za svaki stupac. Na kraju imamo dvije funkcionalnosti “Obrisi” i “uredi” jedna briše sve podatke zaposlenog sa određenim ID druga edituje podatke zaposlenog sa određenim ID. “list” smo dobili iz servleta. Što znači kupili smo sve usere i sačuvali ih u ovu tabelu.

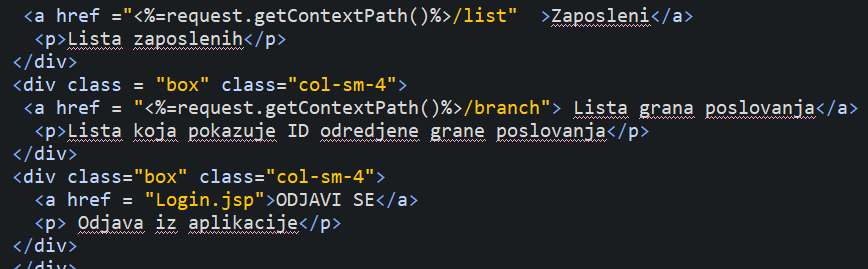
**Branch.jsp**

Je stranica koja jednostavno samo kupi podatke iz određene tabele u bazi podataka i postavlja ih u ovu formu tabele u jsp. Metode za to se takođe nalaze u servletu.

**Login.jsp**

Provjerava unseen podatke u svojoj formi i ako se poklapaju dopušta dalji pristup aplikaciji. Kada se unese korisničko ime i šifra, ako se poklapaju sa onim iz baze podataka proslijedit će nas na sljedeću stranicu “Logged.jsp”

**Logged.jsp**

Ova stranica sadrži tri linka koja vode na određene jsp stranice. 

Prvi void na UserList.jsp koji sadrži glavnu tabelu. Drugi void na stranicu Branch.jsp koja sadrži tabelu grana poslovanja. I treća za odjavu. Odjava na svakoj stranici funkcioniše tako što korisnika samo vrati na prvu stranicu programa tj Login.jsp. samim tim korisnik dok ne unese tačnu šifru i korisničko ime ne može nastaviti dalje.

**Kontakt.jsp**

Kontakt.jsp je jedina stranica u ovom programu koja nije povezana sa servletom ili dao klasom.

<**sql:setDataSource** var = "snapshot" driver = "com.mysql.cj.jdbc.Driver"

url = "jdbc:mysql://localhost/firma?allowPublicKeyRetrieval=true&useSSL=false&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC"

user = "root" password = "123456aa"/>

<**sql:update** dataSource = "${snapshot}" var = "count">

truncate table contact;

</**sql:update**>

<**sql:update** dataSource = "${snapshot}" var = "count">

insert into contact(Firstname,Lastname,Email) select Firstname, Lastname, Email from zaposleni where EmpId>0;

</**sql:update**>

<**sql:query** dataSource = "${snapshot}" var = "result">

select \* from contact;

</**sql:query**>

Ona sadrži logiku u sebi, kao i konekciju. Nakon konekcije stranica prvo briše sve što se nalazi u tabeli. Razlog zbog kojeg prvo briše je to što, ako ne obriše onda će se ponavljati isti podaci. Sa “truncate table” će ih prvo obrisati , pa će da se u bazi podataka upišu ime prezime i email iz tabele zaposleni u tabelu contact, nakon toga se prikazuju svi podaci I upisuju u tabelu na stranici.

1. **Alati i tehnologija korištena pri izradi programa**

* Jsp
* Ide -eclipse
* Jdk
* Apache tomcat server
* Jstl
* Servlet api
* mySql connector
* mySql workbanch