

VISOKA ŠKOLA STRUKOVNIH STUDIJA ARANĐELOVAC



**Dokumentacija – Razvoj aplikacije sa, “na ulogama”
baziranoj kontroli pristupa - Elektronska oglasna tabla**

Profesor:
Filip Vasić

Student:
Marina Kedžić

Arandjelovac, decembar 2019. godine

Rezime

Velika važnost problema zloupotrebljavanja poverljivih informacijama na različitim nivoima u nekim web aplikacijama, dolazi sa tendencijama da se taj proces pristupa ograniči i zaštiti na osnovu određivanja uloga koje će neki korisnik imati. Traži se da se implementacija zaštite što više pojednostavi a takođe i da se efikasnost poveća. Dodatno, ovaj problem je jako složen zbog količine podataka i složenosti samih veza između njih.

U ovom radu biće prikazano jedno rešenje ovog problema i njegova implementacija u programskom jeziku Java uz pomoć korišćenja Spring framework-a koji obezbeđuje zaštitu.

Fokus ove web aplikacije će isključivo biti na pružanju informacija vezanih za uloge koje postoje i njihovim mogućnostima pristupu informacija. Aplikacija omogućava brzo i jednostavno manipulisanje podacima iz baze, koja je deo ove aplikacije, njihov prikaz, izmenu i brisanje.

Ključne reči: dokumentacija, implementacija, automatizacija, programski jezik, Java, Spring Framework, web aplikacija.

SADRŽAJ:

1. Uvod.....	4
1.1. Java programski jezik	4
1.2. NetBeans razvojno okruženje	4
1.3. Spring Framework.	4
1.4. MVC model	5
1.5. MySQL baza podataka.	5
2. Web aplikacija – Elektronska oglasna tabla	6
2.1. Forma za prijavljivanje	6
2.2. Glavne forme	9
2.3. Metode za manipulaciju sa korisnicima.	11
2.3.1. Dodavanje novog nastavnika.....	12
2.3.2. Menjanje postojećeg nastavnika.....	13
2.3.3. Brisanje postojećeg nastavnika.....	14
2.4. Posebni fajlovi	15
3. Baza podataka	16
4. Zaključak	20
5. Reference	21

1. Uvod

U ovom poglavlju će biti opisane tehnologije koje su korišćene u toku razvoja web aplikacije.

1.1. Programski jezik Java

Java¹ objektno-orijentisani programski jezik je originalno kreirao James Gosling iz kompanije „Sun Microsystems“ (koju je 2010. godine kupila korporacija „Oracle“). Cilj je bio da se obezbede jednostavnije i platformski nezavisnije alternative programu C++. Java programi se izvršavaju unutar Java virtuelne mašine, koja je ista na svakoj platformi, posmatrao iz perspektive programera aplikacija. Projekat Java programskog jezika počeo je u junu 1991. godine.

Takođe, važno je napomenuti i da Java spada među case sensitive jezike, koji prave razliku između malih i velikih slova. Osim toga, ovaj programski jezik pripada i grupi takozvanih strongly typed, odnosno strogo tipiziranih jezika, koji zahtevaju eksplicitno identifikovane tipove u kodu.

Java se može naći svuda u: mobilnim uređajima, PDA uređajima, televizorima, u igrama, navigacionim sistemima, desktop aplikacijama, web aplikacijama, RIA(Rich Internet Application)...

1.2. NetBeans IDE razvojno okruženje

NetBeans IDE² vam omogućava brzo i jednostavno razvijanje Java desktop, mobilnih i web aplikacija, kao i HTML5 aplikacija sa HTML-om, JavaScript-om i CSS-om. IDE takođe nudi veliki skup alata za PHP i C / C ++ programera. To je besplatan i otvoreni izvor i ima veliku zajednicu korisnika i programera širom sveta.

IDE je mnogo više od uređivača teksta. NetBeans Editor izdvaja redove, pronalazi ključne reči i ističe izvorni kod - sintaktički i semantički. Omogućava vam jednostavno kucanje koda sa nizom praktičnih i moćnih alata, a takođe nudi i šablone kodova, savete za kodiranje i generatore kodova. NetBeans je programersko okruženje najviše zastupljeno pre svega u svim školama, edukativnim i drugim institucijama, ali i u malim, srednjim i velikim firmama u Srbiji.

1.3. Spring Framework

Spring Framework³ pruža sveobuhvatan model programiranja i konfiguracije za savremene poslovne aplikacije zasnovane na Javi - na bilo kojoj platformi za implementaciju.

Ključni element spring-a je infrastrukturna podrška na nivou aplikacija primer: konfiguriše sigurne procese koje podržavaju niz standarda, protokola, alata; omogućava lakše testiranje; radi sa sistemima za upravljanje relacijskim bazama podataka na Java platformi radi lakšeg upravljanja podacima. U ovom projektu koristili samo neke delove Spring Frameworka kao što su Spring Security – za zaštitu podataka i omogućavanje pristupa i Spring MCV – radi boljeg uvida u aplikaciju i lakšeg rada.

1.4. MCV model

MVC arhitektura⁴ je softverski patern *Model-view-controller* koji odvaja prikaz informacija od interakcije korisnika sa tim informacijama. Model se sastoji od podataka aplikacije, poslovnih pravila, logike i funkcija. View može da bude bilo koji izlazni prikaz podataka, kao što je dijagram ili grafik. Više prikaza istog podatka je moguće, kao što je grafik sa barovima za menadžment i tabelarni prikaz za računovođe. Controller uzima ulazne podatke i konvertuje ih u komande za model ili view. Upotrebljava se u skoro svim programskim jezicima. Model – prikaz – kontroler: HTTP- i servlet-baziran okvir koji pruža mogućnost prilagođavanja za web aplikacije i RESTful (prenos reprezentativnog stanja) web usluga.

MVC “*deli*” aplikaciju na nekoliko nivoa:

1. **Model** – u ovom nivou definišemo klase koje predstavljaju osnovu podataka sa kojima radimo. Ako npr. pravimo aplikaciju za telefonski imenik, u Modelu kreiramo klasu za kontakte koja će pamtit i imena, brojeve telefona itd. (Model nivo u stvari odgovara podacima koji bi se upisivali u bazu podataka) – u našem slučaju view su to Java klase koje su entiteti.
2. **View** – ovde se definiše prikaz podataka iz odgovarajućih modela. View preuzima podatke iz modela, obrađuje ih i izbacuje ih na način koji nam je potreban (npr kombinuje ime i prezime u puno ime i izbaci formatirano po nekim pravilima koje zadam) – u našem slučaju view su HTML stranice.
3. **Controller** – ovde se opisuju izmene, uglavnom inicirane od strane korisnika aplikacije, nad podacima u modelu ili view-u (npr. Dodavanje novih kontakata, njihovo brisanje ili izmena, filtriranje prikaza liste svih kontakata, itd.). – imamo samo jedna java klasa koja ima anotaciju `@controller` i ona kontroliše celu aplikaciju koju smo nazvali `LoginController`.

1.5. MySQL baza podataka

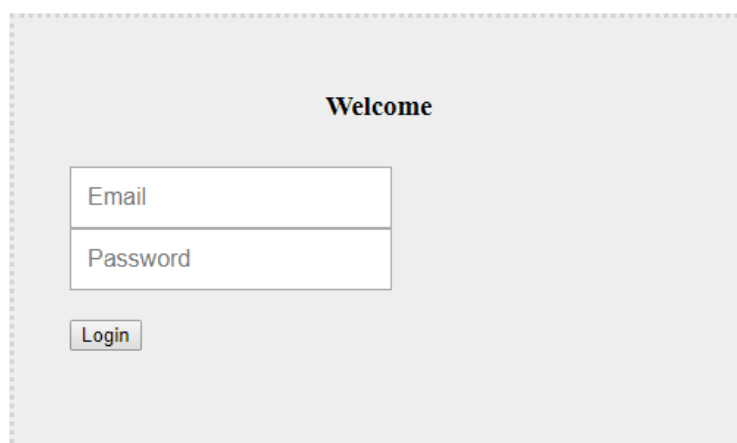
SQL je jezik koji se koristi za pisanje upita unutar **MySQL baze podataka**⁵. Prvobitni vlasnik MySQL-a je bila švedska kompanija MySQL AB, dok je današnji vlasnik Oracle. Prva verzija MySQL baze podataka izašla je 1995. godine. MySQL baza podataka je najpopularnija baza među web aplikacijama i koristi LAMP platformu. Aplikacije koje koriste MySQL bazu podataka su TYPO3, MODx, Joomla, Wordpress, phpBB, MyBB, Drupal kao i mnogi drugi softveri. MySQL takođe koriste i najpoznatiji web sajtovi kao što su Facebook, Twitter, Flickr, Youtube i Google (ne uključujući pretragu). MySQL omogućava pristup bazi podataka uz pomoć većine programskih jezika. MySQL radi na mnogim sistemskim platformama kao što su AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos i Tru64. Portovi MySQL-a prema OpenVMS takođe postoje.

2. Web aplikacija – Elektronska oglasna tabla

Elektronska oglasna tabla je informacijski sistem za poslovanje određene škole koji podržava sve funkcije koje su potrebne u svakodnevnom obaveštavanju roditelja o dešavanjima u školi i dodatno omogućava komunikaciju sa nastavnicima iste škole. U radu se nećemo obazirati na sporedne delove koda već samo na najbitnije delove (samo bitne metode) kojima smo obezbedili mogućnost kontrole pristupa različitim korisnicima sa različitim ulogama. U ovom radu svi kodovi su pisani u programskom jeziku Java.

2.1. Forma za prijavljivanje

Kada se aplikacija pokrene otvara se pretraživač sa formom za prijavu. On zahteva unos korisničkog email-a i lozinke, i izgleda kao na slici 2.1. i njegov kod je prikazan na slici 2.2. Najbitnija metoda u ovoj formi je login() koja se pokreće pritiskom na dugme „Login“, prikazana na slici 2.3., i koja direktno komunicira sa CustomAuthenticationSuccessHandler klasom koja svojom metodom onAuthenticationSuccess(...) obezbeđuje da se u zavisnosti od uloge koju ima određeni korisnik preusmeri na glavnu formu koja odgovara njegovoj ulozi (koja ima posebne funkcije koje ti korisnici mogu da izvršavaju) i ona je prikazana na slici 2.5. Glavne Forme će detaljnije biti opisane u odeljku 2.2.

The image shows a web browser window with a light gray background. At the top, centered, is the word "Welcome" in a bold, black font. Below it, there are two white input boxes stacked vertically. The top box is labeled "Email" and the bottom box is labeled "Password". Below these boxes is a gray button with the word "Login" in white text.

Slika 2.1. Forma za prijavljivanje

```
<body>
<div class="container">
  <form th:action="@{/login}" method="POST" class="form-signin">
    <h3 class="form-signin-heading" th:text="Welcome"></h3>
    <input type="text" id="email" name="email" th:placeholder="Email"
      class="form-control"/> <br/>
    <input type="password" th:placeholder="Password"
      id="password" name="password" class="form-control"/> <br/>
    <div align="center" th:if="${param.error}">
      <p style="font-size: 20; color: #FFC107;">Email or Password invalid, please verify</p>
    </div>
    <button class="btn btn-lg btn-primary btn-block" name="Submit" value="Login" type="Submit"
      th:text="Login"></button>
  </form>
</div>
</body>
```

Slika 2.2. Forma za prijavljivanje kod

```

@RequestMapping(value={"/", "/login"}, method = RequestMethod.GET)
public ModelAndView login(
    @RequestParam(value="error" , required =false) String error ){
    ModelAndView modelAndView = new ModelAndView();
    if(error != null){
        modelAndView.setViewName("error");
    } else
        modelAndView.setViewName("login");
    return modelAndView;
}

```

Slika 2.3. Login() metoda

Metoda configure() – Još jedna jako bitna metoda koja sprečava mogućnost pristupa jedne uloge na više nivoa je metoda configure() koja se nalazi u klasi SecurityConfigurationAdm koja je jedna konfiguraciona klasa koja omogućava web sigurnost. Obeležena je anotacijama @Configuration @EnableWebSecurity koje nam daju na znanje koja je njena uloga. Radi tako što na određene putanje u aplikaciji stavlja dozvolu da im određene uloge mogu pristupiti. (Prikazano je na slici 2.4.). Tako na primer bilo kojoj strani u putanji /parent/** mogu pristupiti samo korisnici čije su uloge ADMIN (administrator) i PARENT (roditelj) – to omogućava hasAnyAuthority() metoda. Dok na primer u putanji /login mogu pristupiti svi korisnici – to omogućava permitAll() metoda. Data metoda i metodom onAuthenticationSuccess(...) koja je prikazana na slici 2.5. koriste Spring Framework-ov security modul koji omogućava zaštitu tako što je importuje na način - import org.springframework.security.*; *bilo koja klasa koju želi da koristi

```

@Override
protected void configure(HttpSecurity http) throws Exception {

    http.
        authorizeRequests()
            .antMatchers("/").permitAll()
            .antMatchers("/login").permitAll()
            .antMatchers("/css/**").permitAll()
            .antMatchers("/images/**").permitAll()
            .antMatchers("/static/**").permitAll()
            .antMatchers("/parent/**").hasAnyAuthority("PARENT", "ADMIN")
            .antMatchers("/teachers/**").hasAnyAuthority("TEACHER", "ADMIN")
            .antMatchers("/admin/**").hasAuthority("ADMIN")
}

```

Slika 2.4. Configure() metoda

```

@Override
public void onAuthenticationSuccess(HttpServletRequest httpServletRequest,
    HttpServletResponse httpServletResponse, Authentication authentication)
    throws IOException, ServletException, RuntimeException
{
    HttpSession session = httpServletRequest.getSession();
    User authUser = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    Admin admin = adminService.findAdminByEmail(authUser.getUsername());

    session.setAttribute("username", authUser.getUsername());

    //set our response to OK status
    httpServletResponse.setStatus(HttpServletResponse.SC_OK);
    Collection<? extends GrantedAuthority> authorities = authentication.getAuthorities();
    authorities.forEach(authority ->
    {
        if(authority.getAuthority().equals("ADMIN"))
        {
            try
            {
                //since we have created our custom success handler, its up to us to where
                //we will redirect the user after successfully login
                httpServletResponse.sendRedirect("/admin/home");
            }
            catch (IOException e)
            {
                throw new RuntimeException(e);
            }
        }
        else if (authority.getAuthority().equals("PARENT"))
        {
            try
            {
                //since we have created our custom success handler, its up to us to where
                //we will redirect the user after successfully login
                httpServletResponse.sendRedirect("/parents/home");
            }
            catch (IOException e)
            {
                throw new RuntimeException(e);
            }
        }
        else if (authority.getAuthority().equals("TEACHER"))
        {
            try
            {
                //since we have created our custom success handler, its up to us to where
                //we will redirect the user after successfully login
                httpServletResponse.sendRedirect("/teachers/home");
            }
            catch (IOException e)
            {
                throw new RuntimeException(e);
            }
        }
    });
}

```

Slika 2.5. onAuthenticationSuccess(...) metoda

2.2. Glavne forme

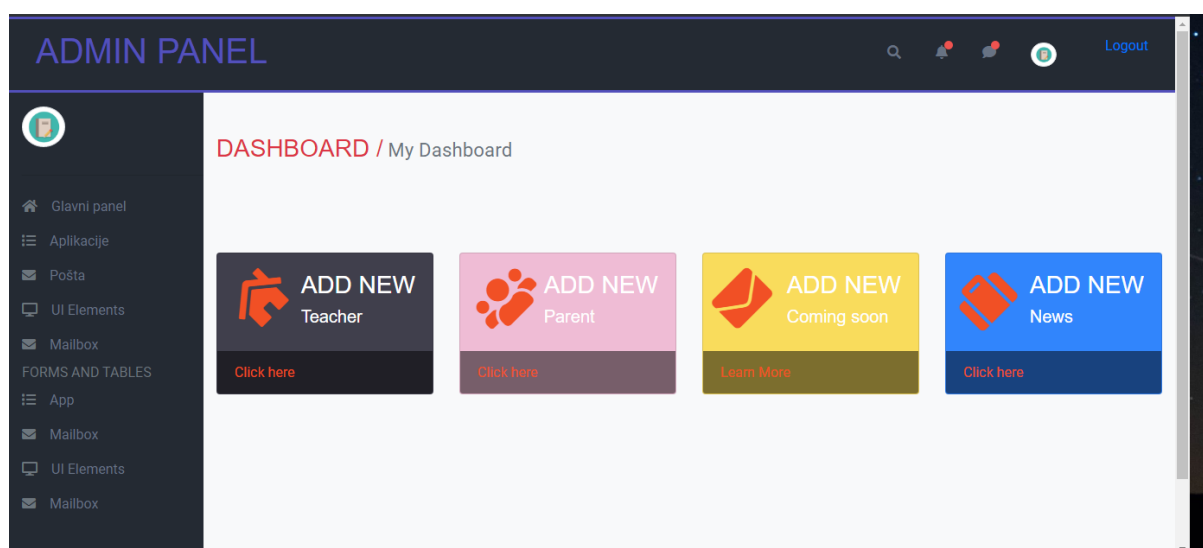
Kada se korisnik uspešno registruje otvara se glavna forma aplikacije u zavisnosti od uloge. Tako da u ovoj aplikaciji postoje uloga administratora, nastavnika i roditelja.

Izgledi glavnih formi prikazani su na slikama 2.6. za administratora, 2.7. za nastavnika i 2.8. za roditelja.

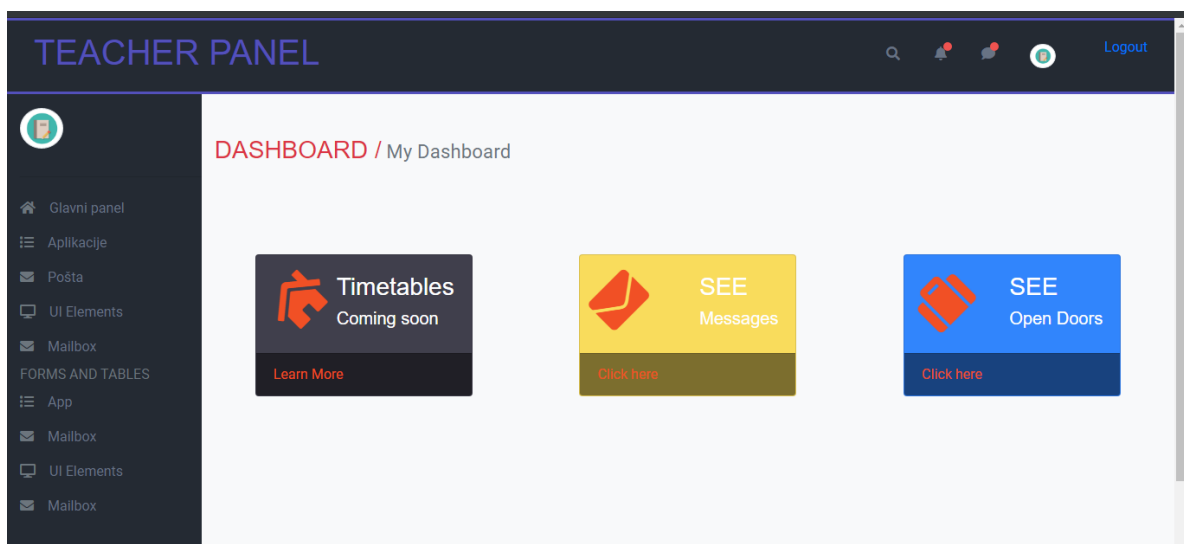
Svaka forma sadrži različite funkcije koje korisnici od zavisnosti od uloge mogu koristiti.

Tako na primer roditelj može poslati poruku učitelju, može da vidi obaveštenja i da zakaže otvorena vrata.

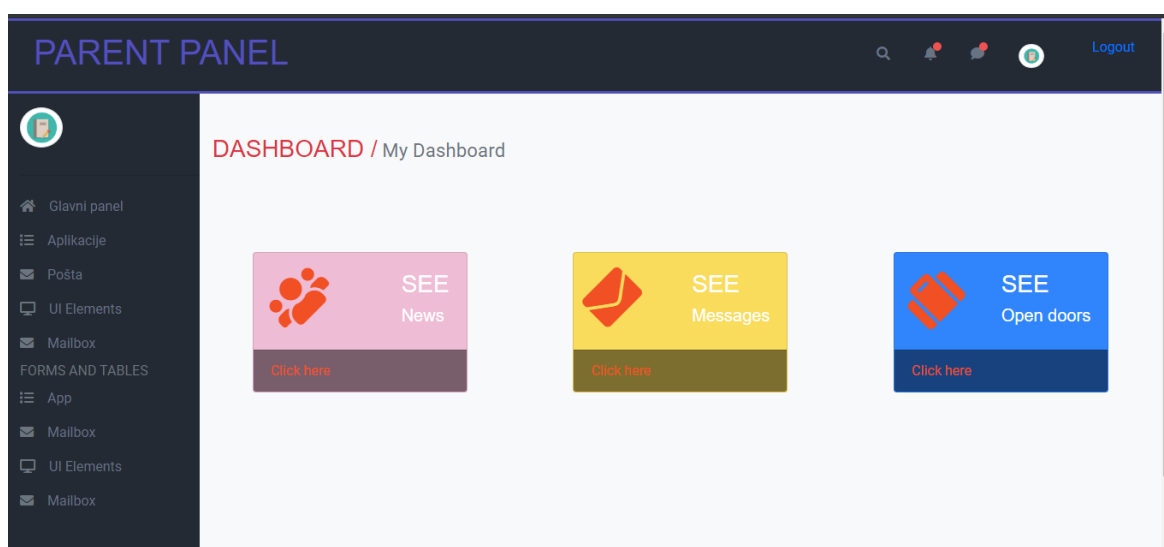
Dok admin može dodavati obaveštenja za roditelje, može dodavati nove korisnike, i to je još jedan način na koji kontrolišemo pristup jer dozvoljavamo jedino administratoru da doda korisnika i da mu on dodeli ulogu, ne postoji sporedno registrovanje što je jako bitno u kontroli pristupa.



Slika 2.6. Glavna forma za administratora



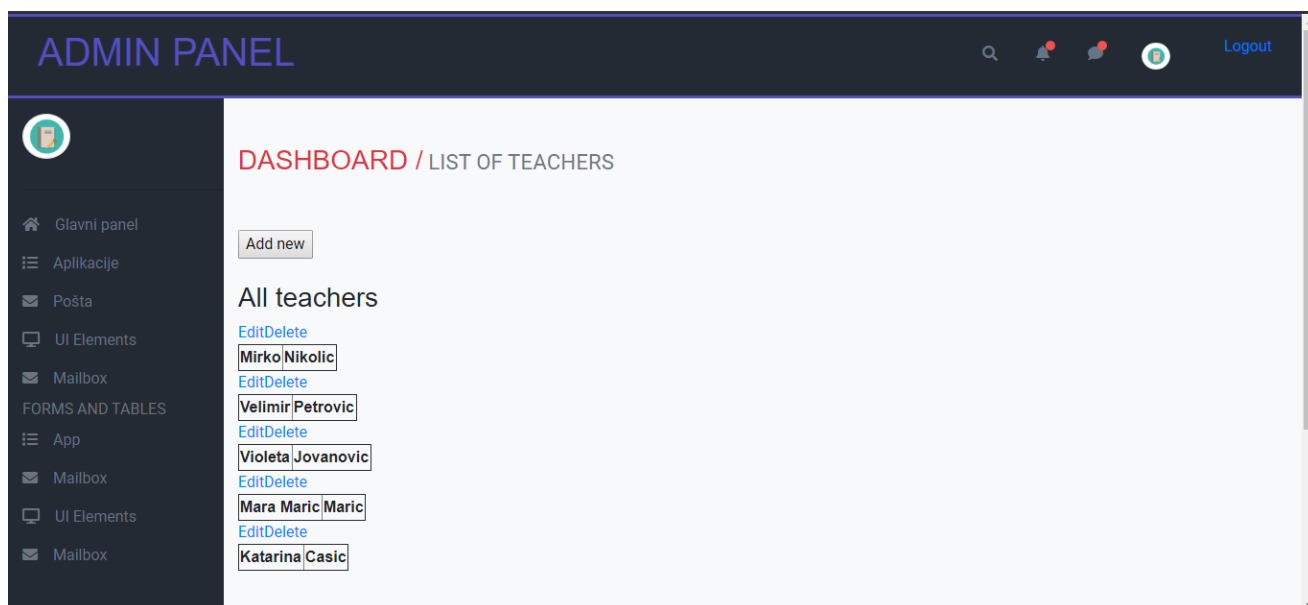
Slika 2.7. Glavna forma za nastavnika



Slika 2.8. Glavna forma za roditelja

2.3. Metode za manipulaciju sa korisnicima

Pošto se ova aplikacija bavi temom kontrole pristupa navešćemo onda samo delove koda (metode) koje se bave time a jedne od najbitnijih su kako to administrator manipuliše korisnicima i bira uloge koje će posedovati. Sada ćemo prikazati koje su mogućnosti koje ima administrator u manipulisanju sa nastavnikom a isti postupak se koristi za manipulisanjem roditelja naravno sa dodeljivanjem različite uloge. Klikom na link „Click here” na prvom kvadratiću sa simbolom kućice otvara se strana na kojoj je prikazana lista svih nastavnika koja se dobija pomoću metode `showTeachers()` prikazanoj na slici 2.10. i iznad imena imamo dva link-a jedan za izmenu nastavnika jedan za brisanje iz baze i dugme „Add new” za dodavanje novog nastavnika a sada dalje razmatraćemo koje se metode koriste za dodavanje izmenu i brisanje jednog nastavnika. Sve ovo je prikazano na slici 2.9.



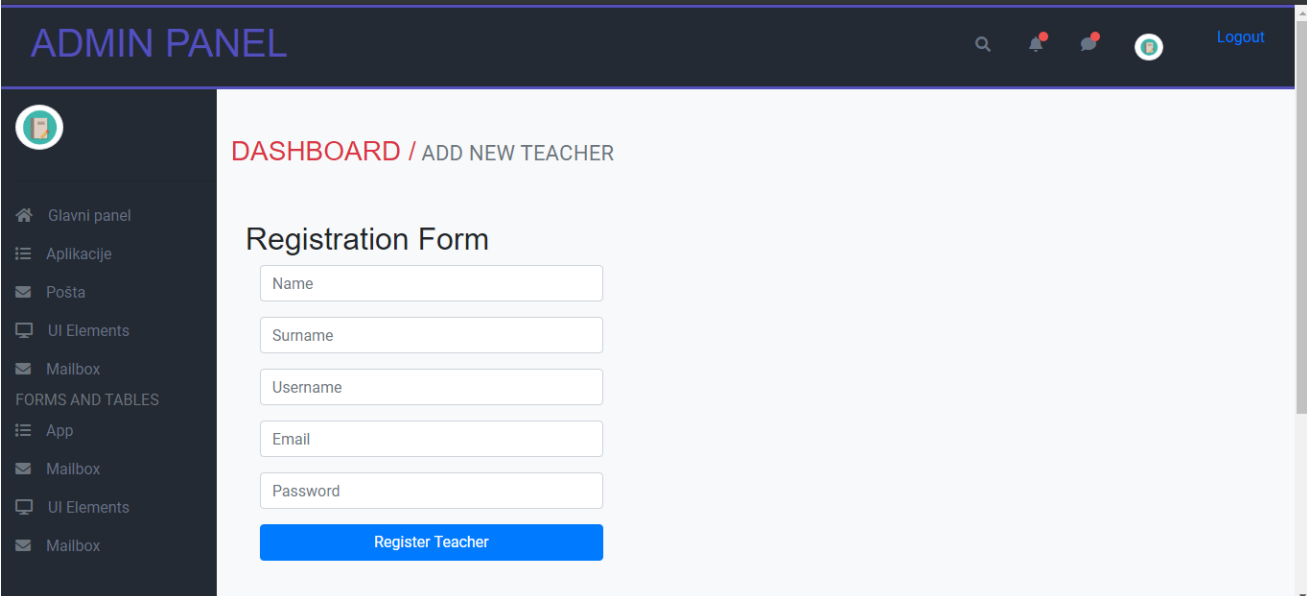
Slika 2.9. –Lista nastavnika

```
@RequestMapping(value="/admin/listteachers", method = RequestMethod.GET)
public ModelAndView showTeachers() {
    ModelAndView modelAndView = new ModelAndView();
    List <Teachers> listanastavnika = teachersServices.findTeachers();
    modelAndView.addObject("listanastavnika", listanastavnika);
    modelAndView.setViewName("/admin/listteachers");
    return modelAndView;
}
```

Slika 2.10. – Metoda `showTeachers()` u `LoginController-u`

2.3.1. Dodavanje novog nastavnika

Klikom na dugme “Add new” otvara se forma za kreiranje novog nastavnika (Prikazana na slici 2.3.1.1.). Za dodavanje koristi se metoda `addNewTeacher()` koja radi po principu da get metodom pravi objekat nastavnika i prosleđuje ga post metodi koja onda popunjava dati objekat (pomoću konstruktora u `Teachers` klasi – prikazanoj na slici 2.3.1.2.) vrednostima iz forme i proverava da li već postoji neki objekat u bazi sa istim email-om. Ako postoji aplikacija javlja da postoje poklapanja i ne upisuje se ništa u bazu. U suprotnom ako ne postoji pomoću metode `saveTeacher()` koja je prikazana na slici 2.3.1.3. i koja se nalazi u `TeachersService` klasi daje mu ulogu nastavnika, šifrira password (kao još jedan način zaštite) i tako pomoću interfejsa `TeachersRepository` i njegovog metoda `save()` čuva objekat u bazu podataka. Metoda `addNewTeacher()` prikazana je na slici 2.3.1.4.



Slika 2.3.1.1. – Forma za kreiranje nastavnika

```
public Teachers(Integer idTeachers, String name, String surname, String username, String password, String email) {
    this.idTeachers = idTeachers;
    this.name = name;
    this.surname = surname;
    this.username = username;
    this.password = password;
    this.email = email;
}
```

Slika 2.3.1.2. – Kontruktor Teachers u Teachers klasi

```
public Teachers saveTeacher(Teachers teachers) {
    teachers.setPassword(bCryptPasswordEncoder.encode(teachers.getPassword()));
    teachers.setActive(3);
    UserRole teachersRole = userroleRepository.findByIdStatus(3);
    teachers.setStatusIdStatus(teachersRole);
    return teachersRepository.save(teachers);
}
```

Slika 2.3.1.3. – Metoda saveTeacher() u TeachersServices klasi

```

@RequestMapping(value="/admin/addNewTeacher", method = RequestMethod.GET)
public ModelAndView addNewTeacher(){
    ModelAndView modelAndView = new ModelAndView();
    Teachers teachers = new Teachers();
    modelAndView.addObject("teachers", teachers);
    return modelAndView;
}

@RequestMapping(value = "/admin/addNewTeacher", method = RequestMethod.POST)
public ModelAndView addNewTeacher(@Valid Teachers teachers, BindingResult bindingResult, ModelMap modelMap) {
    ModelAndView modelAndView = new ModelAndView();
    Teachers teachersExists = teachersServices.findByEmail(teachers.getEmail());
    if (teachersExists != null) {
        bindingResult
            .rejectValue("email", "error.user",
                "There is already a user registered with the email provided");
    }
    if (bindingResult.hasErrors()) {
        modelAndView.setViewName("/admin/novinastavnik");
    } else {
        teachersServices.saveTeacher(teachers);
        modelAndView.addObject("successMessage", "User has been registered successfully");
        modelAndView.setViewName("/admin/addNewTeacher");
    }
    return modelAndView;
}

```

Slika 2.3.1.4. – Metoda addNewTeacher() u LoginController-u

2.3.2. Menjanje postojećeg nastavnika

Klikom na link “edit” otvara se forma za izmenu postojećeg nastavnika (Prikazana na slici 2.3.2.1.) pomoću metode showEditTeachersPage() prikazanoj na slici 2.3.2.2. Za izmenu koristi se metoda editTeacher() koja radi po principu da get metodom pravi objekat nastavnika i prosleđuje ga post metodi koja onda popunjava dati objekat vrednostima iz forme i pritiskom na dugme “Save” menja objekat u bazi podataka. Metoda editTeacher() prikazana je na slici 2.3.2.3.

The screenshot shows a web application interface with a dark sidebar on the left containing navigation links: Glavni panel, Aplikacije, Pošta, UI Elements, Mailbox, FORMS AND TABLES, App, and Mailbox. The main content area has a header 'ADMIN PANEL' and a breadcrumb 'DASHBOARD / EDIT TEACHER'. Below this, the title 'Edit teacher' is displayed. The form contains the following fields: 'Teacher ID' with the value '1', 'Name' with 'Mirko', 'Surname' with 'Nikolic', 'Email' with 'mako', and 'Password' with '\$2a\$10\$ShuMZZ6opV/tcH'. A 'Save' button is located at the bottom of the form.

Slika 2.3.2.1. – Forma za izmenu nastavnika

```

    @RequestMapping("/admin/editteachers/{id}")
    public ModelAndView showEditTeachersPage(@PathVariable(name = "id") Integer id) {
        ModelAndView modelAndView = new ModelAndView("/admin/editteachers");
        Teachers teachers = teachersServices.get(id);
        modelAndView.addObject("teachers", teachers);
        return modelAndView;
    }

```

Slika 2.3.2.2. – Metoda showEditTeachersPage() u LoginController-u

```

    @RequestMapping(value="/admin/editteacher", method = RequestMethod.GET)
    public ModelAndView editTeacher() {
        ModelAndView modelAndView = new ModelAndView();
        Teachers teachers = new Teachers();
        modelAndView.addObject("teachers", teachers);
        return modelAndView;
    }

    @RequestMapping(value = "/admin/editteacher", method = RequestMethod.POST)
    public ModelAndView editTeacher(@Valid Teachers teachers, BindingResult bindingResult, ModelMap modelMap) {
        ModelAndView modelAndView = new ModelAndView();
        teachersServices.saveTeacher(teachers);
        modelAndView.setViewName("/admin/home");
        return modelAndView;
    }

```

Slika 2.3.2.3. – Metoda showEditTeachersPage() u LoginController-u

2.3.3. Brisanje postojećeg nastavnika

Klikom na link “delete” pomoću metode deleteTeachers() prikazanoj na slici 2.3.3.1 objekat se briše iz baze podataka. Tako što koristi metodu delete() koja je prikazana na slici 2.3.3.2. i koja se nalazi u TeachersService klasi i tako pomoću interfejsa TeachersRepository i njegovog metoda delete() briše objekat iz baze podataka.

```

    @RequestMapping("/admin/deleteteachers/{id}")
    public String deleteTeachers(@PathVariable(name = "id") Integer id) {
        teachersServices.delete(id);
        return "redirect:/admin/listteachers";
    }

```

Slika 2.3.3.1. – deleteTeachers() u LoginController-u

```

    public void delete(Integer id) {
        teachersRepository.deleteById(id);
    }

```

Slika 2.3.3.2. – Metoda delete() u TeachersServices klasi

2.4. Posebni fajlovi

Posebni su pošto ih sam korisnik ne može videti jer nemaju formu ali rade jako bitan posao za funkcionisanje same aplikacije.

1. Persistence.xml i application properties

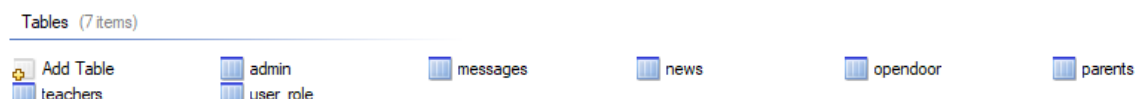
Ovi fajlovi u ovom projektu konfigurišu konekciju sa bazom i omogućavaju olakšan rad sa podacima sa kojom ce se manipulirati tokom rada same aplikacije. Plus korišćenje nekih tehnologija koje olakšavaju pravljenje aplikacije u ovoj aplikaciji korišćen je Thymeleaf koji pomaže lakšem komuniciranju između HTML strana i Java programskog jezika.

2. DemoApplication.java

Ova java klasa u sebi ima main() metod koja pokreće celu aplikaciju u sebi sadrži run() metodu koja pokreće Spring aplikacije a ova je jedna od njih poput ove.

3. Baza podataka

Ova baza je rađena u MySQL-u. Sadrži 7 tabela. Nazivi tabela mogu se videti na slici 3.1. Tabela „user_role” je jedna od najbitnijih jer ona sadrži uloge koje neki korisnik može da ima pa na osnovu nje možemo da kontrolišemo pristup aplikaciji pa ćemo se najviše na njoj zadržati.



Slika 3.1. Struktura baze










Struktura tabela su prikazane na sledećim slikama a pokušaćemo jednu da objasnimo da bi ostale bile jasne. Evo na primer tabelu „teachers” Struktura tabele „clanovi” je prikazana na Slici 3.3. Tabela ima 8 kolona: „id_teachers“, „name“, „surname“, „username“, „password“, „status_id_status“ – spoljašnji ključ povezan sa „user_role” tabelom i određuje koju ulogu poseduje, „active“ i „email“. „id_teachers“ je primaran ključ (PK) i jedinstveni. NN – znači da ne smeju da ostanu prazni a UQ – znači da je jedinstven.

teachers - Table										
Table Name: teachers		Schema: e_diary								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_teachers	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
surname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(245)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
status_id_status	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
active	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	





Slika 3.3. Struktura tabele „teachers”

admin - Table										
Table Name: admin		Schema: e_diary								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
surname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(235)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
status_id_status	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
active	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
email	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL







Slika 3.2. Struktura tabele „admin”

parents - Table x										
 Table Name: <input type="text" value="parents"/> Schema: e_diary										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_parents	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 surname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(235)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 status_id_status	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 active	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 email	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	







Slika 3.4. Struktura tabele „parents“

news - Table x										
 Table Name: <input type="text" value="news"/> Schema: e_diary										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_news	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 messages	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 date_and_time	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Slika 3.5. Struktura tabele „news“

messages - Table x										
 Table Name: <input type="text" value="messages"/> Schema: e_diary										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_messages	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 text_messages	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 time	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 message_id_teacher	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 message_id_parent	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Slika 3.6. Struktura tabele „messages“



opendoor - Table x										
 Table Name: <input type="text" value="opendoor"/> Schema: e_diary										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_opendoor	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 date_and_time	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 teachers_id_teachers	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 parents_id_parents	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 active	TINYINT(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Slika 3.7. Struktura tabele „opendoor“

Struktura tabele “user_role” je prikazana na Slici 3.8. A ono što se nalazi u njoj na slici 3.9. Iz tabele možemo da zaključimo da korisnici koje kao status imaju broj 1 znači da su admini, sa 2 da su direktori, sa 3 da su nastavnici i sa 4 da su roditelji.

user_role - Table ×

Table Name: Schema: **e_diary**

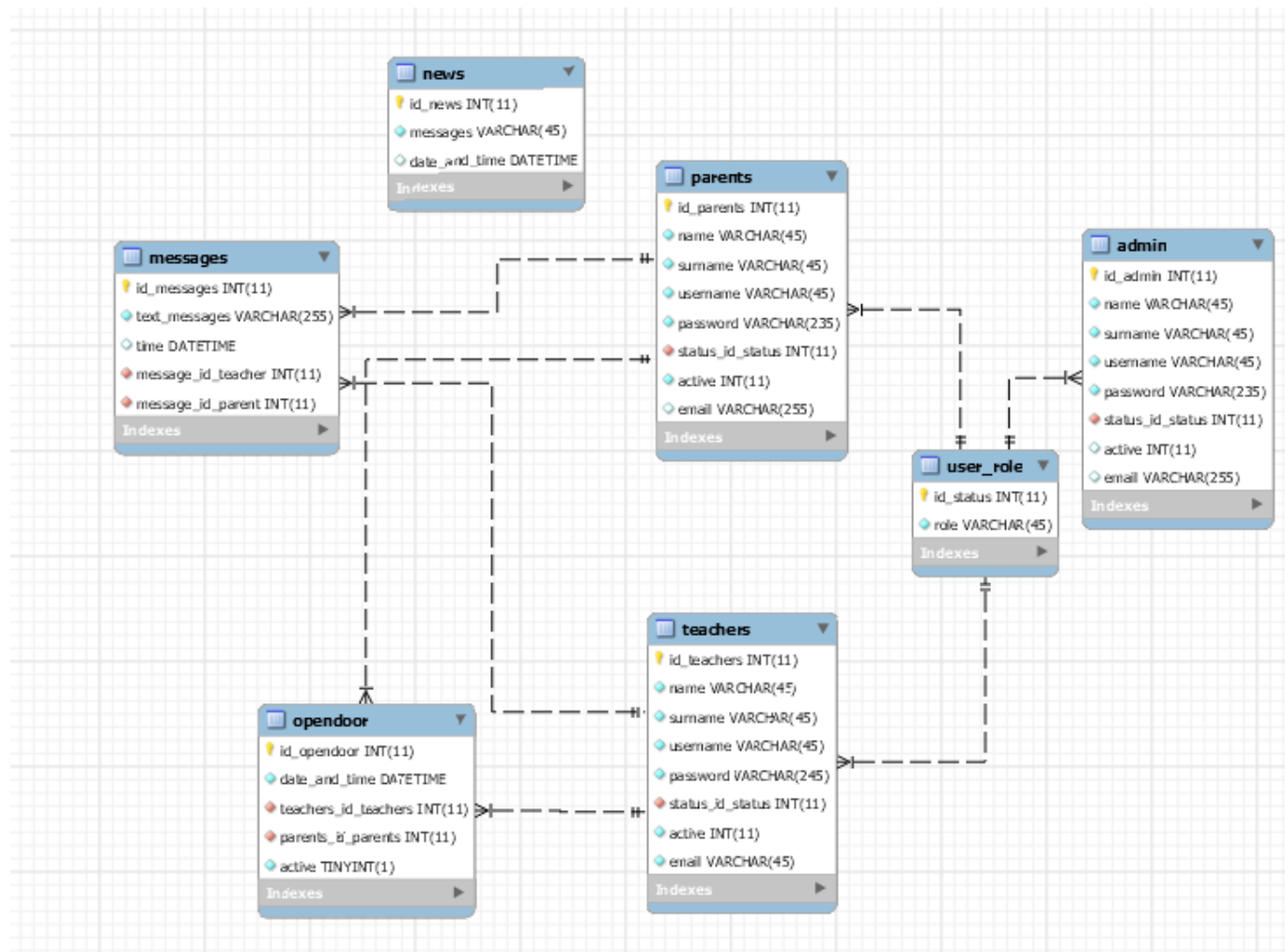
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_status	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 role	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Slika 3.8. Struktura tabele „user_role“

	id_status	role
▶	1	ADMIN
	2	HEADMASTER
	3	TEACHER
	4	PARENT
✱	NULL	NULL

Slika 3.9. Podaci u tabeli „user_role“

Predstavljeno grafički, možemo videti veze između tabela i da utvrdimo u kakvom su odnosu radi lakšeg razumevanja same aplikacije.



Slika 3.10. Relacija između tabela u bazi

4. Zaključak

U radu smo predstavili web aplikaciju koja obezbeđuje kontrolisan pristup po ulogama. Za buduće unapređenje web aplikacije može se dodati još neke određene uloge ili čak napraviti elektronski dnevnik. Može se uvesti i komunikacija između admina i ostalih korisnika radi unapređenja aplikacije dobijanjem feedback-ova korisničkih iskustva.

A kao još bolju zaštitu mogao bi da se koristi neki bolji algoritam za šifriranje password-a. Jer što više podataka enkriptujemo teško će zlonamerni hakeri uspeti da preuzmu podatke i da ih koriste na loš način.

I kao napomenu za sada je pristup aplikaciji omogućen samo jednom adminu. Ako postoje potrebe za korišćenje aplikacije od strane drugih admina, može se uvesti da ih ima više.

5. Reference

1. Java,

Yakov Fain (2015) „*Java Programming Second Edition*“, Wiley Publishing, Inc.

2. NetBeans IDE,

<https://netbeans.org/>

(Pristupljeno 02.12.2019. godine)

3. Spring Framework

<https://spring.io/projects/spring-security>

(Pristupljeno 02.12.2019. godine)

4. MVC arhitektura (Model),

<https://blog.petrovic.gr/sr/2016/12/vodic-za-ucenje-web-programiranja-frontend-pristup/#MVC>

(Pristupljeno 02.12.2019. godine)

5. MySQL database,

<https://www.mysql.com/>

(Pristupljeno 02.12.2019. godine)