Web aplikacija za prezentiranje društvenih događaja

Proskura, Rudolf

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:672913

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-09**

Repository / Repozitorij:

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

RUDOLF PROSKURA

WEB APLIKACIJA ZA PREZENTIRANJE DRUŠTVENIH DOGAĐAJA

ZAVRŠNI RAD

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

RUDOLF PROSKURA

WEB APLIKACIJA ZA PREZENTIRANJE DRUŠTVENIH DOGAĐAJA

WEB APPLICATION FOR PRESENTING SOCIAL EVENTS

ZAVRŠNI RAD

Mentor:

dr. sc. Sanja Brekalo prof. v. š

ČAKOVEC, 2022.

ZAHVALA

Zahvaljujem mentorici dr.sc. Sanji Brekalo na pomoći i savjetima pri izradi ovoga rada. Zahvaljujem svojim roditeljima, majci Mariji i ocu Igoru te svojemu bratu Bartolu na potpori tijekom mojega studiranja i na tome što su uvijek vjerovali u mene.

Rudolf Proskura

SAŽETAK

Tema je ovoga završnog rada izrada web-aplikacije za prezentiranje društvenih događaja. Aplikacija omogućuje upravljanje društvenim događajima – omogućuje unos, pretraživanje i ažuriranje događaja te registraciju korisnika i uređivanje korisničkih podataka. Također, aplikacija je namijenjena svim korisnicima koje zanima neki događaj te korisnicima koji sami žele organizirati, odnosno prezentirati neki događaj.

Za potrebe rada korištene su Java tehnologije. Te su tehnologije CSS (*engl. Cascading Style Sheets*), PrimeFaces, JSF (*engl. Java ServerFaces*) te JavaScript i HTML (*engl. Hyper Text Markup Language*). Tehnologije kao što su CSS, PrimeFaces te JSF i HTML koriste se za izradu sučelja web-stranice, kao što su dinamičnost stranice, forme za upis teksta i padajućih izbornika, postavljanje slika, oblik teksta, veličina slova. Prednost je *PrimeFacesa* u tome da sadrži već gotov kod za pojedine forme. JavaScript upravlja navedenim tehnologijama te ih objedinjuje u jednu funkcionalnu cjelinu. Da bi se to sve povezalo, moraju se povezati CSS, JSF, HTML datoteke s datotekom unutar oznake <head> s glavnim dijelom programa. Za pohranu podataka korištene su MySql baza podataka i LDAP direktorij. Programski kod pisan je u integriranome razvojnom okruženju IntelliJ IDEA. Primarna je namjena aplikacije jednostavan pristup informacijama i upravljanje informacijama o društvenim događajima koji su podijeljeni u kategorije. Kod odabira kategorije korisnik može odabrati jednu kategoriju za događaj koji želi prezentirati. Postoje četiri razine korisnika aplikacije, a razina određuje mogućnosti koje korisnik ima u aplikaciji. Te su razine korisnika aplikacije gost, registrirani korisnik, organizator i administrator. Kod registracije korisnika na stranicu vrši se validacija polja za unos podataka. Ukoliko prigodom unosa dođe do pogreške, javlja se poruka da je došlo do pogreške u određenom dijelu prijave. Administrator ima apsolutnu kontrolu nad aplikacijom i nad upravljanjem podatcima korisnika. To uključuje mijenjanje uloga registriranim korisnicima, brisanje korisnika, brisanje i ažuriranje događaja drugih korisnika.

U radu su detaljnije opisane mogućnosti aplikacije, način na koji je ona izrađena te su opisane korištene tehnologije.

***Ključne riječi**: aplikacija, Java, prezentiranje događaja, JavaScript, LDAP*

Sadržaj

1. Uvod 7

2. Cilj rada 8

3. Arhitektura sustava i korištene tehnologije 9

3.1. Troslojna arhitektura 9

3.2. Korištene tehnologije i primjeri 10

3.2.1. Prezentacijski sloj 12

MVC uzorak dizajna 12

JSF i Primefaces 13

MVC primjer iz aplikacije 14

View 14

3.2.2. Sloj poslovne logike 15

3.2.3. Sloj pristupa podatcima 19

3.3. Razvojno okruženje 27

3.3.1. Aplikacijski poslužitelj 27

3.3.2. Baza podataka 27

3.3.3. Apache Directory Studio 29

3.3.4. IntelliJ IDEA 30

4. Aplikacija 32

4.1. Struktura projekta 32

4.2. Model baze podataka 36

4.3. Uloge i korisnici aplikacije 38

4.3.1 Gost 38

4.3.2 Registrirani korisnik 40

4.3.3 Organizator 42

4.3.3 Administrator 43

4.4. Opcije i mogućnosti aplikacije 45

5.Zaključak 52

6. Popis literature 53

Prilozi 54

Popis slika 54

1. Uvod

Java je jedna od popularnijih razvojnih platforma koja ima široku primjenu u razvoju informacijskih sustava. Koristi se u razvoju različitih programskih rješenja, od poslužitelja aplikacija, desktop aplikacija, web-aplikacija do aplikacija za mobilne uređaje te pametne kartice. Java platforma dostupna je u svim operacijskim sustavima (Windows, Unix, Linux, Mac) i potpuno je prenosiva između njih. Značajno je obilježje Java programskoga jezika WORA način (engl. *write once, run anywhere*) koji označava mogućnosti upisivanja Java koda jednom, a on se zatim kompajliran može izvršavati na bilo kojoj platformi koja podržava Java programski jezik[1].

Aplikacija „Online Events“ zamišljena je kao alat za upravljanje društvenim događajima, za njihovu prezentaciju te za dostupnost informacija širokoj publici.

2. Cilj rada

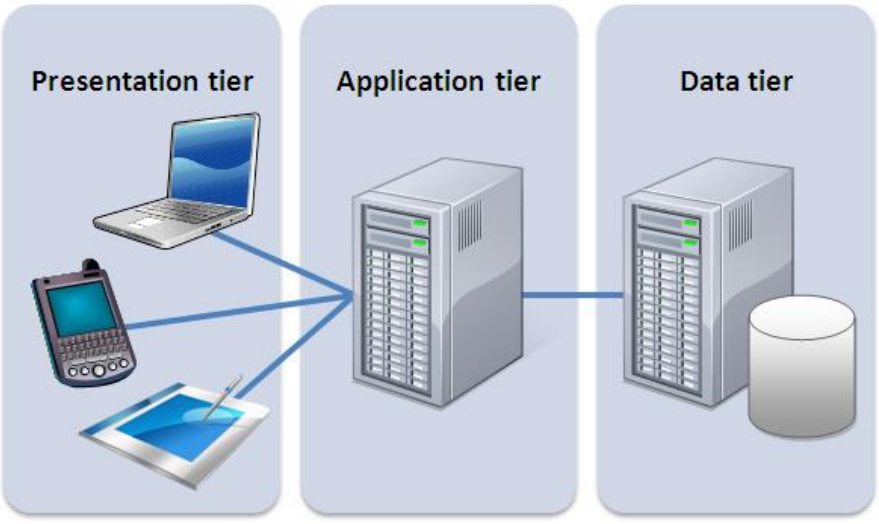
Cilj je ovoga završnog rada predstaviti razvoj web-aplikacija Java tehnologijama na primjeru aplikacije za prezentiranje društvenih događaja. Motivacija za odabir teme i odabir korištenih tehnologija proizlaze iz činjenice da u današnje vrijeme postoji velika potreba za razvojem takvih vrsta aplikacija, a Java tehnologija jedna je od vodećih u ovome području. Krajnji je cilj aplikacije mogućnost prezentiranja društvenih događaja kako bi se olakšalo njihovo prezentiranje na organiziran, jednostavan i korisniku zanimljiv način.

3. Arhitektura sustava i korištene tehnologije

3.1. Troslojna arhitektura

Softverska arhitektura struktura je informacijskoga sustava koja se sastoji od entiteta, njihovih vanjskih vidljivih karakteristika te veza između njih. Drugim riječima, softverska arhitektura definira se kao opis podsustava i komponenti informacijskoga sustava te veza između njih.[2]

Postoji mnogo podjela i vrsta programskih arhitektura, a za web-aplikaciju završnoga rada korištena je višeslojna arhitektura (često se naziva n-tier arhitektura) – konkretno troslojna arhitektura (slika1). Troslojna arhitektura jedna je od najčešće korištenih arhitektura kod tradicionalnih klijent-poslužitelj aplikacija. Kao što samo ime govori, sastoji se od triju slojeva: prezentacijskoga sloja, sloja poslovne logike (često se naziva aplikacijskim slojem, srednjim slojem) i podatkovnoga sloja (odnosno sloja pristupa podatcima).



Slika 1. Troslojna arhitektura
IZVOR: https://managementmania.com/en/three-tier-architecture

Prezentacijski sloj predstavlja korisničko sučelje i komunikacijski sloj aplikacije koji služi za interakciju korisnika s aplikacijom. Glavna mu je svrha prikaz podataka korisnicima i prikupljanje podataka od korisnika.

Sloj poslovne logike (aplikacijski sloj) srednji je sloj i predstavlja „srce“ aplikacije. Zadužen je za obradu prikupljenih podataka i podataka iz podatkovnoga sloja. U ovom

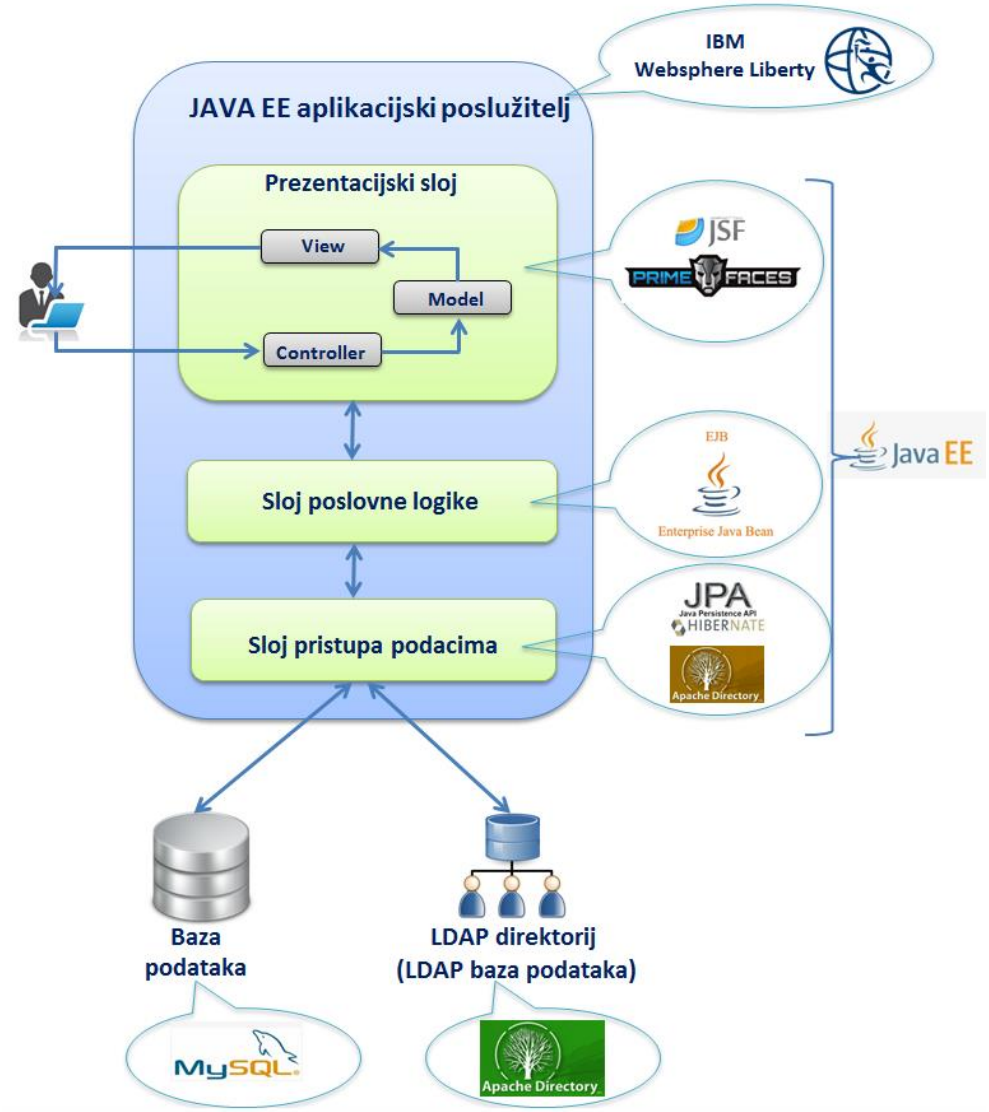
sloju implementirana su poslovna pravila te poslovna logika, koji predstavljaju glavne funkcije aplikacije.
Podatkovni sloj (sloj pristupa podatcima) dio je u kojemu se upravlja operacijama čitanja, upisa podataka u bazu i brisanja podataka iz baze.

Prednosti su troslojne arhitekture sljedeće [3]:

- Poboljšana skalabilnost – aplikacijski poslužitelj može biti naseljen na više fizičkih poslužitelja. Također, baza podataka ne uspostavlja vezu sa svakim klijentom – potrebna je veza s manjim brojem aplikacijskih poslužitelja.
- Poboljšan integritet podataka – u ovoj arhitekturi svi ažurirani podaci idu kroz srednji sloj (sloj poslovne logike). Uklonjen je rizik nepouzdanih klijentskih aplikacija koje oštećuju (korumpiraju) podatke.
- Poboljšana je sigurnost jer klijenti nemaju izravan pristup bazi podataka; klijenti mnogo teže mogu dohvatiti neautorizirane podatke. Poslovna je logika sigurnija jer je pohranjena na centralnom i sigurnom poslužitelju.
- Uravnoteženje opterećenja mnogo je lakše razdvajanjem poslovne logike od poslužitelja baze podataka.
- Održavanje i izmjene mnogo su lakše. Može se odvojeno upravljati kodom prezentacijskoga sloja i sloja poslovne logike na način da jedan sloj ne utječe na drugi.
- Poboljšana je ponovna/višestruka upotreba.

3.2. Korištene tehnologije i primjeri

U ovom dijelu ukratko su predstavljene korištene tehnologije podijeljene po slojevima prethodno opisane troslojne arhitekture i ostali sastavni dijelovi sustava u kojem je aplikacija implementirana (slika 2). Cijeli sustav temelji se na Java programskom jeziku i Java tehnologijama.



Slika 2. Arhitektura i korištene tehnologije

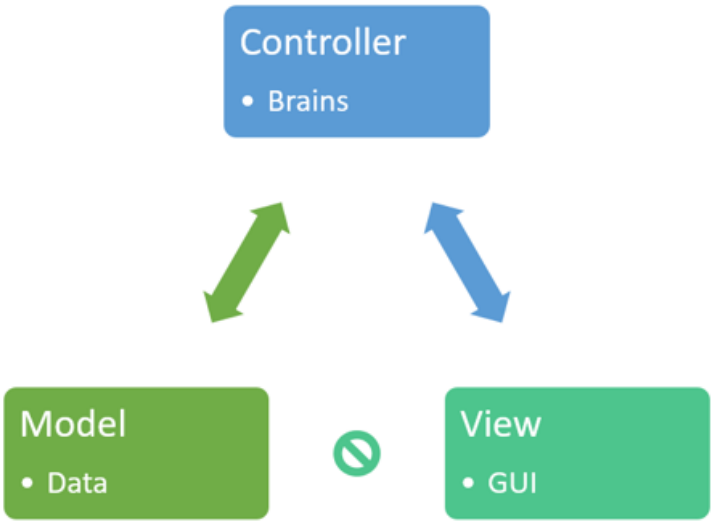
Izvor: https://www.ibm.com/cloud/learn/three-tier-architecture

3.2.1. Prezentacijski sloj

MVC uzorak dizajna

U prezentacijskom sloju koristi se Model View Controller (MVC) uzorak dizajna. MVC se uglavnom odnosi na korisničko sučelje, prezentacijski sloj aplikacije, odnosno sloj aplikacije koji služi za interakciju s korisnikom. Ima tri glavne komponente koje su logički odvojene: model, view i controller, što je prikazano na slici 3. Svaka komponenta ima svoju zadaću:

- Model sadrži samo „čiste“ podatke s kojima korisnik radi. To mogu biti podatci koji se prenose između view i controller komponente, a mogu biti i podatci povezani s poslovnom logikom. Npr. objekt DogadajDto u aplikaciji završnoga rada dohvatit će podatke o događaju iz sloja poslovne logike (koji dohvaća podatke iz baze podataka preko sloja pristupa podatcima) i koristit će ih za prikaz podataka korisniku.
- View prikazuje podatke modela korisniku. Ova komponenta pristupa podatcima i prikazuje ih. Npr. događaj view uključuje sve UI (User Interface) komponente poput unosa teksta, padajućih izbornika koji služe za interakciju s korisnikom.
- Controller povezuje model i view komponentu, odnosno predstavlja sučelje između modela i viewa za procesiranje poslovne logike i ulaznih zahtjeva, upravlja podatcima koristeći model komponentu i komunicira s view komponentom da bi se podatci prikazali.



Slika 3. Komponente MVC uzorka dizajna
Izvor: https://www.educative.io/blog/mvc-tutorial

JSF i Primefaces

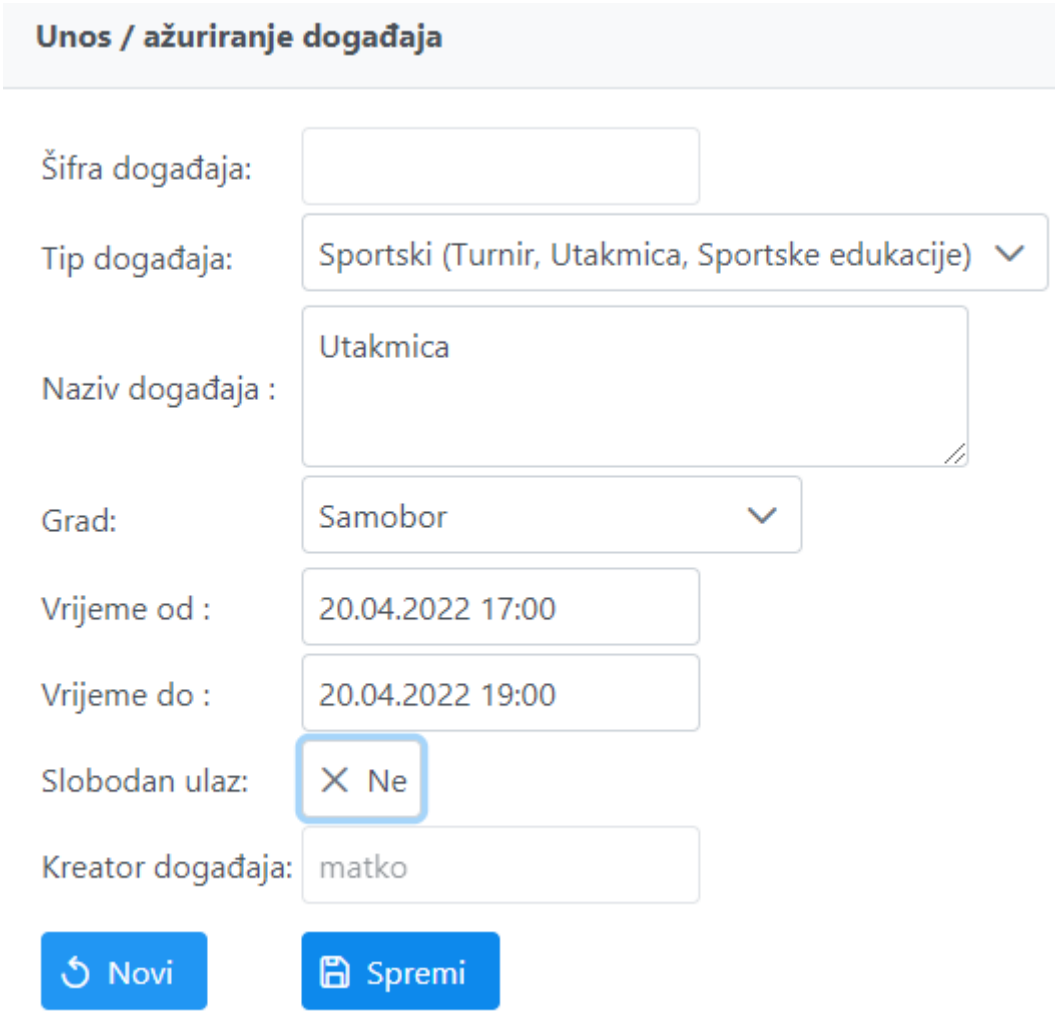
JSF (JavaServer Faces) i Primefaces glavne su tehnologije koje se koriste u prezentacijskome sloju. JSF (JavaServerFaces) pripada standardnim Java tehnologijama za izradu web-sučelja temeljnih na komponentama i orijentiranima na događaje te prati MVC uzorak dizajna. Glavna je ideja i prednost razvojnih okvira poput JSF-a enkapsulacija klijentskih tehnologija poput HTML-a, CSS-a i JavaScripta koji programerima omogućuju izradu web-sučelja s minimalnim doticajem s tim tehnologijama.

JSF predstavlja XML dokument u kojemu su UI (User Interface) komponente iza kojih su Java objekti. Za UI komponente koristio se Primefaces. Primefaces je open source biblioteka UI komponenti za JSF implementirana u Java programski jezik. Koristi se u velikim organizacijama, sadrži široku paletu UI komponenti, jednostavno se koristi i dobro je dokumentirana.

MVC primjer iz aplikacije

Ovdje je prikazan konkretan primjer MVC uzorka iz aplikacije.

View



Slika 4. Dio korisničkoga sučelja/ekrana na opciji Unos / Ažuriranje događaja
Izvor: Autor

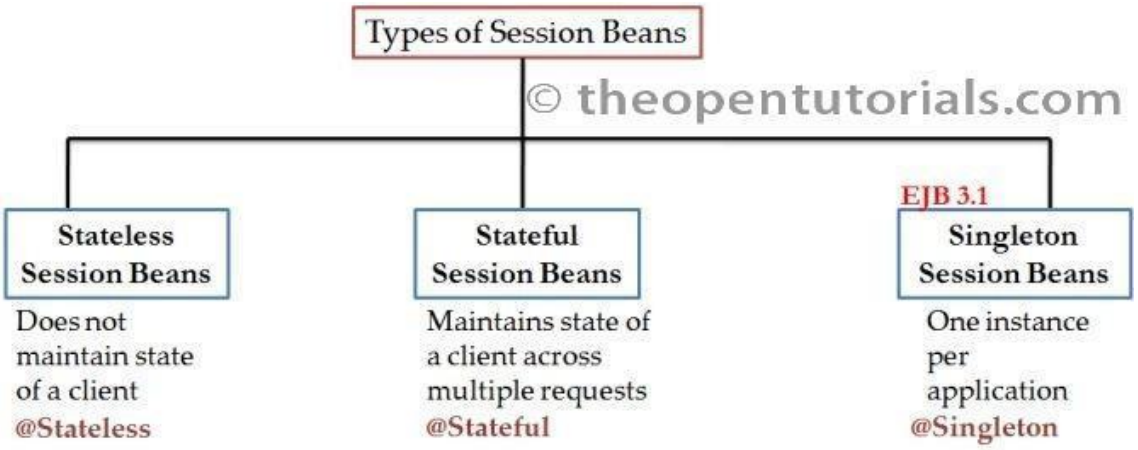
Na slici 4. prikazan je dio korisničkoga sučelja ili ekrana, a u MVC-u on je zapravo .xhtml datoteka koja se sastoji od gotovih UI komponenti: inputText, inputTextarea, selectOneMenu, calendar.

3.2.2. Sloj poslovne logike

Sloj poslovne logike (aplikacijski sloj) srednji je sloj i predstavlja „srce“ aplikacije. Zadužen je za obradu prikupljenih podataka iz prezentacijskoga sloja i podataka iz podatkovnoga sloja. U ovom sloju implementirana su poslovna pravila i poslovna logika koje predstavljaju glavne funkcije aplikacije. Drugim riječima, sloj poslovne logike predstavlja sučelje između prezentacijskoga sloja i sloja pristupa podataka te je u njemu implementirana sva poslovna logika (npr. validacija podataka, obračuni). Također, u sklopu ovoga sloja definira se ispravan redoslijed odvijanja aktivnosti potrebnih da bi se završila određena akcija ili zadatak.

U projektu su za sloj poslovne logike korištene Entreprise JavaBeans komponente (EJBs), a mogu se koristiti i druge Java EE tehnologije kao što su JAX-RS RESTful web-servisi, JAX-WS web-servisi, Java Persistence API entiteti. Enterprise JavaBean (EJBs) komponente su koje implementiraju poslovnu logiku i prema tome pripadaju sloju poslovne logike u višeslojnim aplikacijama. Njima upravlja komponenta aplikacijskoga servera koja se zove EJB kontejner. Između ostaloga, EJB kontejner daje Enterprise JavaBeans komponentama pristup važnim servisima poput onih za upravljanje transakcijama i sigurnosnim mehanizmima.

Postoje tri vrste Enterprise JavaBeanova: Session Bean, Entity Bean i Message-Driven Bean. U sloju poslovne logike korišten je Session Bean, konkretno Stateless Session Bean.

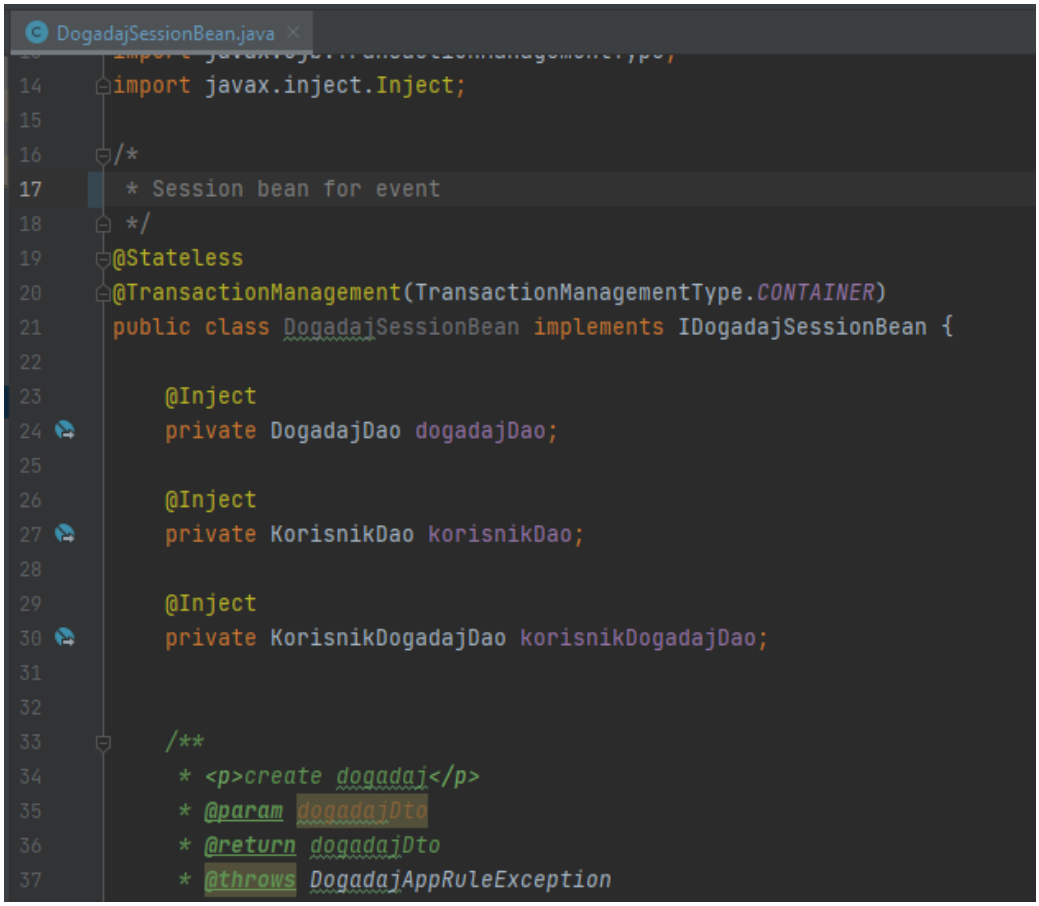


Slika 5. Vrste Session Beana
Izvor: https://theopentutorials.com/tutorials/java-ee/ejb3/session-beans/types-of-session-beans/

Razlikujemo tri vrste Session Beana [4]:

- Stateful: Ova vrsta Session Beana održava informacije o stanju komunikacije između poslužitelja i klijenta. To znači sljedeće: kad EJB klijent dohvati instancu ovog tipa beana, vrijednosti koje se odnose na ovu instancu ostaju konzistentne tijekom komunikacije. Stateful Beanovi jedinstveni su za pojedinoga klijenta, oni predstavljaju stanje klijenta.
- Stateless: Ova vrsta Session Beana prilično je jednostavnija jer ne zahtijeva spremanje informacije o stanju komunikacije za klijenta. Prema tome, Stateless Beanovi nemaju nikakvo stanje i kao takve dijeli ih više klijenata.
- Singleton: Ova vrsta Session Beana kreirana je jednom na nivou aplikacije. Kao rezultat ona održava informacije o stanju između poziva klijenta tijekom cijeloga života aplikacije.

Na slici 6. jedan je primjer EJB Java klase iz projekta – DogadajSessionBean, konkretno Stateless Session Bean. U odnosu na POJO (Plain Old Java Object) mogu se prepoznati po @Stateless anotaciji. @TransactionManagement anotacija definira način upravljanja transakcijama u Session Beanu. U ovom slučaju EJB kontejner upravlja transakcijama.



Slika 6. Primjer Stateless Session Beana
Izvor: Autor

Kao što je prije spomenuto, u sloj poslovne logike implementirana je poslovna logika, a slike 7. i 8. prikazuju konkretne primjere iz projekta u kojima se provodi validacija obaveznih podataka za događaj i korisnika.

```
private void validateBeforeCreate(DogadajDto dogadajDto) throws DogadajAppRuleException {
    boolean hasError = false;
    List<String> messages = new ArrayList<~>();

    if (dogadajDto == null) {
        hasError = true;
        messages.add("Događaj nema podatke!");
    }
    if (dogadajDto.getTipDogadaja() == null) {
        hasError = true;
        messages.add("Događaj nama odabran tip!");
    }
    if (StringUtils.isBlank(dogadajDto.getNazivDogadaja())) {
        hasError = true;
        messages.add("Događaj nema popunjen naziv!");
    }
    if (dogadajDto.getVrijemeOd() == null) {
        hasError = true;
        messages.add("Događaj nema odabrano vrijeme od!");
    }
    if (dogadajDto.getVrijemeDo() == null) {
        hasError = true;
        messages.add("Događaj nema odabrano vrijeme do!");
    }
    if (dogadajDto.getVrijemeOd() != null && dogadajDto.getVrijemeOd().isBefore(LocalDateTime.now())) {
        hasError = true;
        messages.add("Događaj ne može biti u proplosti!");
    }
```

Slika 7. Primjer validacije podataka za događaj prije samoga spremanja u bazu podataka
Izvor: Autor

```
public void validateBeforeCreate(KorisnikDto korisnikDto) throws DogadajAppRuleException {
    boolean hasError = false;
    List<String> messages = new ArrayList<~>();

    if (korisnikDto == null) {
        hasError = true;
        messages.add("Korisnik nema podatke!");
    }
    if (StringUtils.isBlank(korisnikDto.getKorisnickoIme())) {
        hasError = true;
        messages.add("Korisničko ime je obavezan podatak!");
    }
    if (StringUtils.isNotBlank(korisnikDto.getKorisnickoIme()) && (korisnikDto.getKorisnickoIme().length() < 3 ||
            korisnikDto.getKorisnickoIme().length() > 20)) {
        hasError = true;
        messages.add("Korisničko ime mora imati minimalno 3 znaka, a maksimalno 20 znakova!");
    }
    if (StringUtils.isNotBlank(korisnikDto.getKorisnickoIme()) && korisnikDto.getKorisnickoIme().contains(" ")) {
        hasError = true;
        messages.add("Korisničko ime ne smije sadržavati razmak!");
    }
    if (StringUtils.isNotBlank(korisnikDto.getKorisnickoIme()) && checkUserNamesExists(korisnikDto.getKorisnickoIme())) {
        hasError = true;
        messages.add("Korisničko ime " + korisnikDto.getKorisnickoIme() + " već se koristi!");
    }
    if (StringUtils.isBlank(korisnikDto.getIme())) {
        hasError = true;
        messages.add("Ime je obavezan podatak!");
```

Slika 8. Primjer validacije podataka za korisnika prije samog spremanja u bazu podataka
Izvor: Autor

Također, u sklopu ovoga sloja definira se ispravan redoslijed odvijanja aktivnosti potrebnih da bi se završila određena akcija ili zadatak, npr. ukoliko se briše korisnika iz sustava, potrebno je odraditi sljedeće aktivnosti: izbrisati događaje koje korisnik ima u svojemu kalendaru, izbrisati događaje koje je korisnik kreirao iz kalendara drugih korisnika, izbrisati događaje koje je korisnik kreirao te na kraju izbrisati korisnika (slika 9).

```
public void deleteKorisnik(String korisnik) throws DogadajAppRuleException {
    //DB delete dogadaj korisnik
    korisnikDogadajDao.deleteKorisnikDogadajByKorisnik(korisnik);
    //DB delete dogadaj korisnik
    korisnikDogadajDao.deleteKorisnikDogadajByKreator(korisnik);
    //DB delete dogadaj
    korisnikDogadajDao.deleteDogadajByCreator(korisnik);
    //DB delete korisnik
    korisnikDao.deleteKorisnik(korisnik);
    //LDAP delete
    korisnikDao.deleteUserFromLDAP(korisnik);
}
```

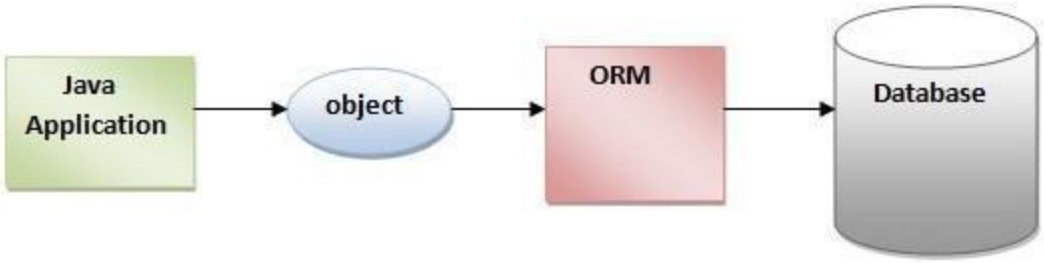
Slika 9. Primjer redoslijeda akcije prigodom brisanja korisnika
Izvor: Autor

3.2.3. Sloj pristupa podatcima

Sloj pristupa podatcima ili kako se često naziva podatkovni sloj ima ulogu odvajanja prezentacijskoga sloja, sloja poslovne logike od podataka. Koristi se za operacije čitanja, pisanja, brisanja podataka iz različitih spremišta podataka / u različita spremišta podataka (u ovom projektu to su baza podataka i LDAP direktorij ili LDAP baza podataka). Postoje dva glavna razloga zašto se koristiti sloj pristupa podatcima. Prvi je razlog apstrakcija baze podataka ili drugih izvora podataka. Na taj način može se mnogo jednostavnije napraviti migracija s npr. MySQL-a na Oracle bazu podataka. Drugi razlog je da možemo napraviti promjene u sloju poslovne logike bez ikakva utjecaja na fizički model podataka. U svemu tome pomažu ORM (Object Relational Mapping) mehanizmi. Osim ORM-a u sloju pristupa podatcima koristio se i Apache Directory LDAP API.

3.2.3.1 ORM, Hibernate, JPA

ORM je proces konvertiranja Java objekata u tablice baze podataka. Drugim riječima, ORM omogućuje interakciju s relacijskom bazom podataka bez SQL upita.



Slika 10. Veza između sloja poslovne logike i baze podataka (ORM)
Izvor: https://www.javatpoint.com/hibernate-tutorial

Ovdje još treba spomenuti JPA (Java Persistenece API), specifikaciju koja definira funkcionalnosti i standarde za ORM mehanizme. U projektu koristimo Hibernate koji je jedan od najpopularnijih Java razvojnih okvira. Hibernate je standardna implementacija JPA specifikacije. Razvoj sloja pristupa podatcima u Hibernate razvojnom okviru temelji se na anotacijama. Na sljedećoj je slici primjer JPA entiteta u kojemu je korištena Hibernate anotacija. Na početku se kreira jednostavna Java klasa Dogadaj koja predstavlja podatke o događaju koji se sprema u bazu podataka. Da bi Hibernate mehanizam to prepoznao, mora se definirati JPA entitet na način da iznad naziva Java klase dodamo anotaciju @Entity. Osim @Entity anotacije koristimo sljedeće JPA anotacije:

- @Inheritance – definira strategiju nasljeđivanja klasa u odnosu na tablice baze podataka.
- @Table - definira tablicu i shemu na koji se entitet odnosi.
- @Id – označava primarni ključ tablice.
- @GeneratedValue – definira način generiranja identifikatora, odnosno vrijednosti primarnoga ključa.
- @Column – određuje kolonu tablice, odnosno naziv kolone tablice.
- @JoinColumn i @OneToOne – kombinacija ovih dviju anotacija govori da se kolona u tom entitetu odnosi na primarni ključ entiteta na koji se referencira.

Dogadaj.java

```java
/**
 *
 * Dogadaj entity class
 *
 */
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@Table(name = "dogadaj", schema = "online_events")
public class Dogadaj {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "sifra", nullable = false)
    private Integer sifraDogadaja;

    @Column(name = "naziv", nullable = false, length = 250)
    private String nazivDogadaja;

    @Column(name = "vrijeme_od")
    private LocalDateTime vrijemeOd;

    @Column(name = "vrijeme_do")
    private LocalDateTime vrijemeDo;

    @JoinColumn(name = "grad", referencedColumnName = "sifra")
    @ManyToOne
    private Grad grad;
```

Slika 11. JPA entitet Dogadaj
Izvor: Autor

Važan je dio JPA-a EntityManager koji implementira programsko sučelje i pravila definirana JPA specifikacijom. EntityManager daje mogućnost interakcije s bazom podataka koristeći metode koje implementira. Na slici 12. prikazan je primjer korištenja EntityManager metoda:

- persist() – Koristi se kod kreiranja/umetanja novih slogova u bazi podataka.
- merge() – Koristi se za izmjenu postojećih objekata. Merge() metoda traži objekt u bazi s istim id-om ako ga nađe, ažurira ga, a ako ga ne nađe, kreira novi.
- flush() – Nakon poziva ove metode promjene su odmah vidljive u bazi podataka, ne ovise o tome je li transakcija završila ili ne.

Ukoliko se za upis podataka ne koristi EntityManager, treba otvoriti vezu prema bazi podataka, izvršiti insert SQL upit te zatvoriti vezu prema bazi. Može se zaključiti da

korištenje EntityManagera i njegovih metoda olakšava razvoj i smanjuje količinu koda i mogućnost pogrešaka.

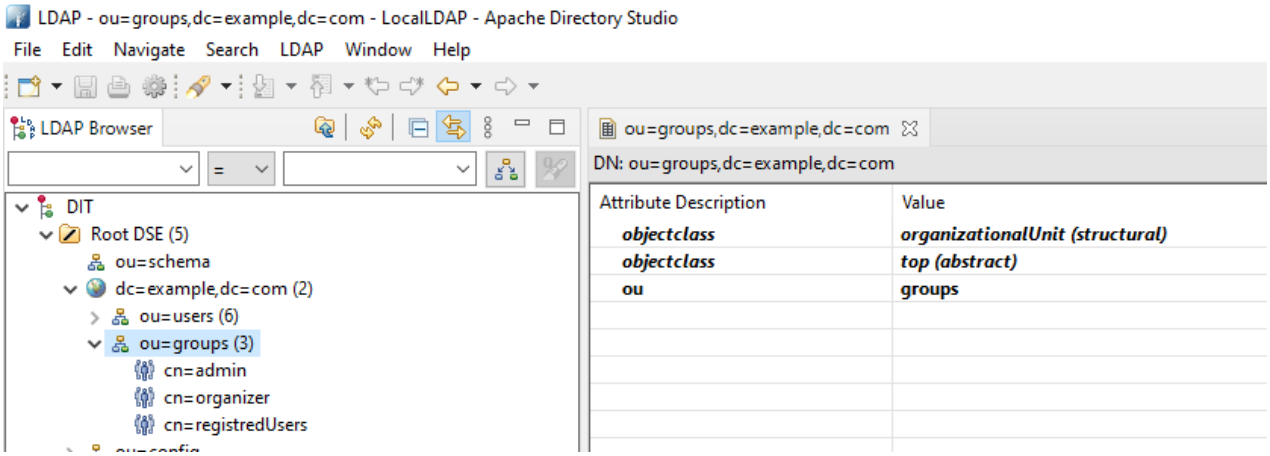
```
/**
 * Create new dogadaj entity
 *
 * @param dto
 * @return
 * @throws DogadajAppRuleException
 */
public DogadajDto create(DogadajDto dto) throws DogadajAppRuleException {
    validateBeforeCreate(dto);
    Dogadaj entity = formEntity(dto);
    getEntityManager().persist(entity);
    getEntityManager().flush();
    return formDTO(entity);
}

/**
 * Edit existing dogadaj entity
 *
 * @param dogadajDto
 * @throws DogadajAppRuleException
 */
public void edit(DogadajDto dogadajDto, String logedUser) throws DogadajAppRuleException {
    validateBeforeEdit(dogadajDto, logedUser);
    Dogadaj entity = formEntity(dogadajDto);
    getEntityManager().merge(entity);
    getEntityManager().flush();
}
```

Slika 12. Primjer korištenja EntityManager metoda u projektu
Izvor: Autor

3.2.3.2 Apache Directory LDAP API

Podatci o korisnicima aplikacije spremaju se u Apache DS. LDAP (Lightweight Directory Access Protocol) jedan je od glavnih protokola koji je razvijen za servise direktorija (procese sigurnog upravljanja korisnicima i njihovim pravima pristupa IT resursima). U biti, LDAP specificira način pohrane podataka u direktorij te olakšava autentifikaciju i autorizaciju korisnika na poslužitelje, datoteke, mrežnu opremu i aplikacije. [5]
Apache DS (Directory Server) ugradiv je, proširiv, usklađen sa standardima, moderan LDAP poslužitelj implementiran u Java programski jezik i dostupan pod Apache softverskom licencom. Podržava i druge mrežne protokole kao što su Kerberos i NTP, ali u osnovi je to LDAP direktorij poslužitelj. Ugradiv znači da ga je moguće konfigurirati,

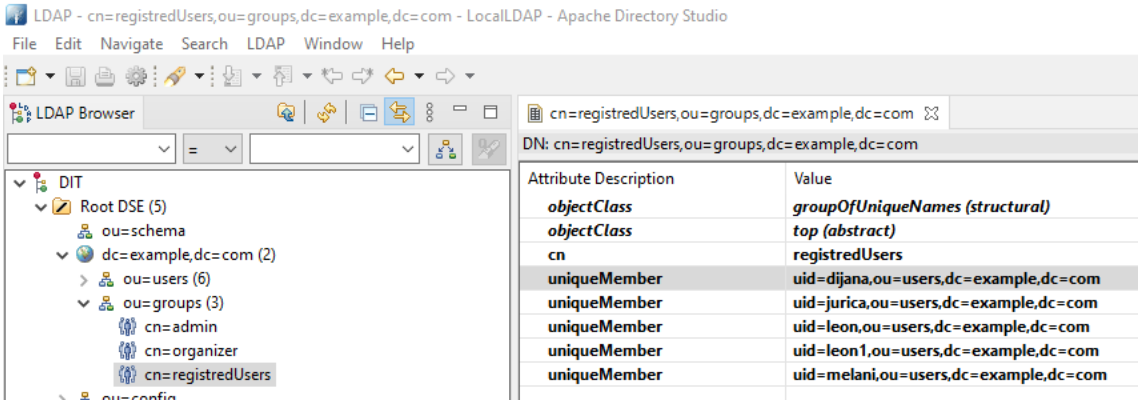
pokrenuti, zaustaviti iz drugih Java komponenti, osobito aplikacijskih poslužitelja. Proširiv znači da moderna arhitektura rješenja pruža mnoge točke proširenja [6]. Podatci u LDAP poslužitelju organizirani su u hijerarhijsko-relacijskom formatu. Hijerarhijski je zato što svaki zapis, osim korijenskoga, ima jedan "roditeljski" zapis, a relacijski jer se više zapisa može grupirati zajedno. Najviša razina hijerarhije u LDAP poslužitelju naziva se domenom. U jednom takvom poslužitelju može postojati više domena. Ispod domene su grane koje predstavljaju organizacijske jedinice koje su najčešće odjeli neke organizacije. Za potrebe ovoga projekta kreirana je domena example.com s jednostavnom strukturom u kojoj postoje tri organizacijske jedinice, odnosno grupe korisnika: admin, organizer, registeredUsers.



Slika13. Grupe LDAP direktorija

Izvor: Autor

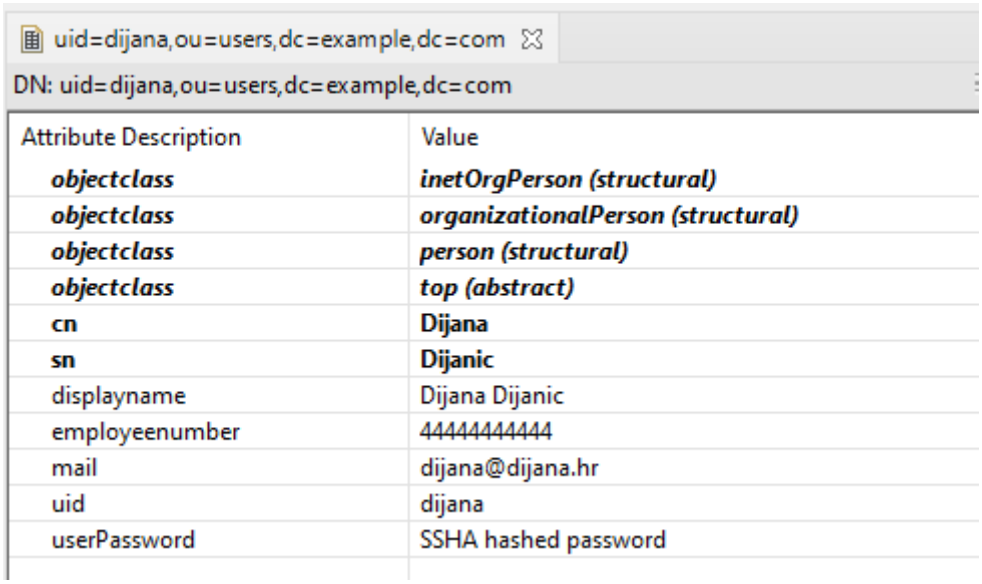
Grupa se sastoji od korisnika, što se vidi na slici 15. U konkretnom su primjeru u grupi registredUsers korisnici dijana, jurica, leon, leon1 i melani.



Slika 15. Korisnici LDAP grupe registredUsers

Izvor: Autor

Svaki korisnik ima atribute, a konkretan primjer prikazan je sljedećom slikom.



uid=dijana,ou=users,dc=example,dc=com

DN: uid=dijana,ou=users,dc=example,dc=com

Attribute Description	Value
objectclass	***inetOrgPerson (structural)***
objectclass	***organizationalPerson (structural)***
objectclass	***person (structural)***
objectclass	***top (abstract)***
cn	**Dijana**
sn	**Dijanic**
displayname	Dijana Dijanic
employeenumber	44444444444
mail	dijana@dijana.hr
uid	dijana
userPassword	SSHA hashed password

Slika 14. Pregled atributa LDAP korisnika

Izvor: Autor

S obzirom na to da se u sklopu projekta za spremanje podataka o korisnicima aplikacije, za autentifikaciju i autorizaciju koristi Apache DS, potreban je način na koji se može komunicirati između aplikacije i LDAP poslužitelja (Apache DS-a), način kako doći do podataka koji su u LDAP direktoriju pohranjeni. To je omogućio Apache Directory LDAP API – skup gotovih metoda i mehanizama za komunikaciju s LDAP poslužiteljem. Iz Apache Directory LDAP API-a korištene su sljedeće metode ili operacije:

- otvaranje veze – Prije nego se mogu dohvatiti podatci iz LDAP direktorija, treba uspostaviti, odnosno otvoriti vezu prema poslužitelju LDAP direktorija.
  LdapConnection connection = new LdapNetworkConnection ("localhost", 389);
  Ovo je primjer otvaranja veze prema lokalnom poslužitelju LDAP direktorija koristeći port 389.
- zatvaranje veze – Kad veza više nije potrebna, zatvara se jednostavnom naredbom close().
  connection.close();
- otvaranje sjednice – Nakon otvaranje veze potrebno je otvoriti sjednicu koja će čuvati korisničke podatke tijekom sjednice.
  ```
  connection.bind( "uid=admin,ou=system", "secret" );
  ```

- pretraga – koristi se kad treba dohvatiti korisnike prema određenom kriteriju, npr. želimo pregled svih korisnika koji su organizatori ili sve korisnike koji se zovu Marko.
- dodavanje, brisanje, ažuriranje zapisa.

U nastavku je prikaz nekih operacija korištenih u projektu.

Slika 16. prikazuje dio metode u kojoj se radi dodavanje novoga korisnika. Koristi se add metoda u koju šaljemo korisničko ime, prezime, OIB, lozinku i adresu e-pošte.

Slika 16. Dodavanje novoga korisnika

Izvor: Autor

Na slici 17. može se vidjeti kako se vrši pretraga korisnika po korisničkome imenu metodom search.

```
Dn userNameDN = new Dn( ...upRdns: "uid=" + userName + ",ou=users,dc=example,dc=com"); //nade odredenog usera
EntryCursor cursor = connection.search(userNameDN,  s: "(objectclass=*)", SearchScope.OBJECT);

for (Entry entry : cursor) {
    korisnikDto = new KorisnikDto();
    korisnikDto.setKorisnickoIme(userName);
    korisnikDto.setIme(entry.get("cn").get().toString());
    korisnikDto.setPrezime(entry.get("sn").get().toString());
    korisnikDto.setEmail(entry.get("mail").get().toString());
    korisnikDto.setOib(entry.get("employeenumber").get().toString());
    korisnikDto.setLozinka((entry.get("userpassword").get().toString()));
    //    System.out.println(entry);
}
```

Slika 17. Pretraga korisnika prema korisničkome imenu i punjenje objekta KorisnikDto podatcima korisnika

Izvor: Autor

Ovdje na slici 18. prikazana je metoda koja omogućuje korisniku promjenu adrese e-pošte.

```
Modification replaceMail = new DefaultModification(ModificationOperation.REPLACE_ATTRIBUTE,  attributeId: "mail",
        korisnikDto.getEmail());
connection.modify( s: "uid=" + korisnikDto.getKorisnickoIme() + ",ou=users,dc=example,dc=com", replaceMail);
```

Slika 18. Promjena adrese e-pošte korisnika

Izvor: Autor

Slika 19. prikazuje metodu za premještanje korisnika iz jedne grupe u drugu.

```
if (!StringUtils.equals(postojecaGrupa, korisnikDto.getTipKorisnika())) {
    //add to new group

    Modification addUniqueMember = new DefaultModification(ModificationOperation.ADD_ATTRIBUTE,
            attributeId: "uniqueMember",  ...values: "uid=" + korisnikDto.getKorisnickoIme() + ",ou=users,dc=example,dc=com");
    connection.modify( s: "cn=" + korisnikDto.getTipKorisnika() + ",ou=groups,dc=example,dc=com", addUniqueMember);

    //remove from current group
    Modification removeUniqueMember = new DefaultModification(ModificationOperation.REMOVE_ATTRIBUTE,
            attributeId: "uniqueMember",  ...values: "uid=" + korisnikDto.getKorisnickoIme() + ",ou=users,dc=example,dc=com");
    connection.modify( s: "cn=" + postojecaGrupa + ",ou=groups,dc=example,dc=com", removeUniqueMember);
}
```

Slika 19. Premještanje korisnika iz jedne u drugu grupu – promjena tipa korisnika

Izvor: Autor

.

3.3. Razvojno okruženje

3.3.1. Aplikacijski poslužitelj

Za aplikacijski poslužitelj odabran je IBM WebSphere Liberty. Radi se o brzom i dinamičnom Java aplikacijskom poslužitelju koji kombinira IBM tehnologije s open-source softverom. Navedeni poslužitelj podržava mikroservise i modele programiranja zasnovane na standardima osmišljenima na način da se programeri moderniziraju osobnim tempom. IBM WebSphere Liberty vrlo je jednostavan za korištenje, postoje dobre upute za konfiguraciju i ima dosta dokumentacije koja pomaže prigodom korištenja. Osim jednostavnosti korištenja i brzog učenja IBM aplikacijski poslužitelj ima mnogo referenci, koristi se u ozbiljnim i velikim organizacijama, što je također bio jedan od razloga zašto je odabran taj aplikacijski poslužitelj. Sama aplikacija može se naseliti na bilo koji Java aplikacijski poslužitelj, ali uz prethodnu konfiguraciju aplikacijskoga poslužitelja.

3.3.2. Baza podataka

U projektu se koristila MySQL baza podataka. Također, kao i aplikacijski poslužitelj, baza podataka može se lako zamijeniti i podatci se mogu migrirati na druge baze podataka. MySQL je odabran zbog toga jer je jedan od popularnijih sustava za upravljanje bazama podataka, a ne treba plaćati licencu za korištenje, otvorenog je koda.

U nazivu "MySQL", SQL dio znači "strukturirani jezik upita". SQL je najčešći standardizirani jezik koji se koristi za pristup bazama podataka. SQL je definiran ANSI / ISO SQL standardom. Osnovne su SQL naredbe koje se najčešće koriste sljedeće:

SELECT - Koristi se za dohvaćanje podataka.

INSERT - Služi za umetanje novih podataka.

UPDATE - Koristi se za ažuriranje podataka koji već postoje.

DELETE - Koristi se za brisanje podataka.

CREATE - Koristi se za stvaranje nove tablice u bazi.

ALTER - Koristi se za ažuriranje postojeće tablice.

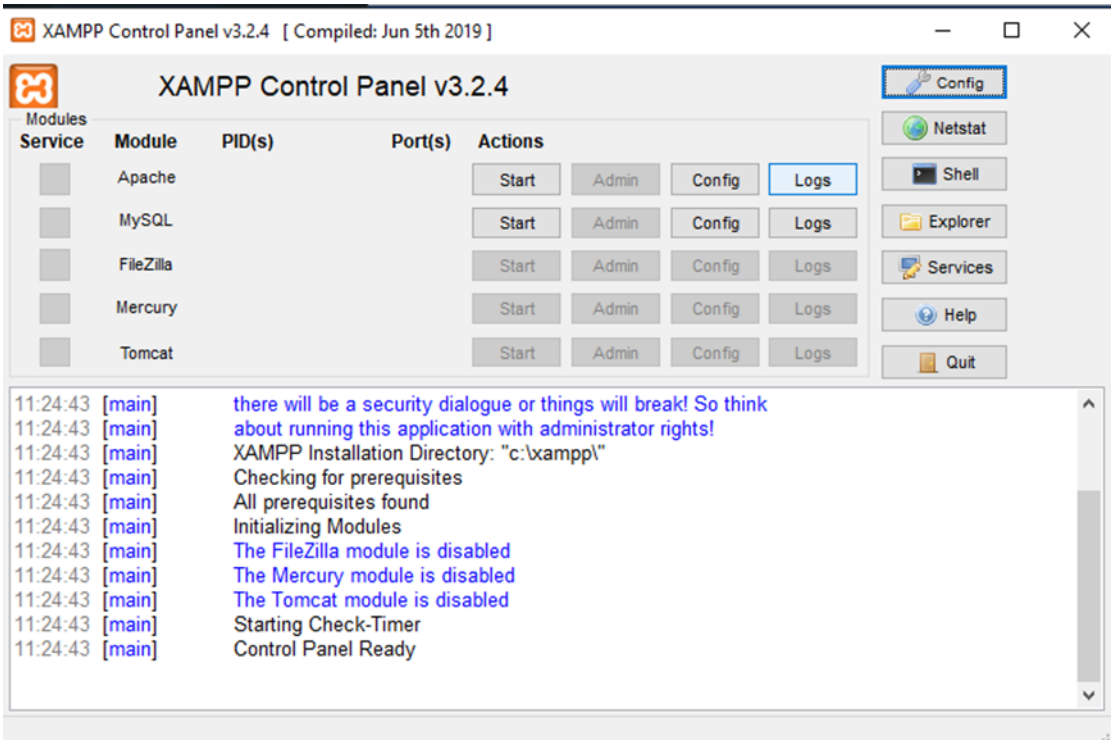
DROP - Koristi se za brisanje tablice.

Od alata još su korišteni MySQL Workbench i XAMPP.

MySQL Workbench objedinjeni je vizualni alat za arhitekte baza podataka, programere i administratore baza podataka. MySQL omogućuje modeliranje podataka, SQL razvoj i objedinjene administracijske alate za konfiguraciju servera, korisnika, sigurnosnih kopija i mnogo više. Dostupan je na Windowsima, Linuxu i maxOS operativnom sustavu [7]. U sklopu projekta ovaj alat koristio se uglavnom za izvršavanje SQL upita kao što su kreiranje tablica, upis šifranika, a često i za dohvat podataka tijekom izrade aplikacije kako bi se moglo provjeriti jesu li svi podatci točno upisani, ažurirani ili izbrisani.

-

XAMPP je jedna od najčešće korištenih platformi web-poslužitelja koja pomaže programerima da razvijaju i testiraju svoje programe na lokalnom poslužitelju. Sastoji se od Apache HTTP poslužitelja, MySQL poslužitelja (novije verzije imaju MariaDB) i interpretera za različite jezike kao što su PHP i Perl. XAMPP je dostupan na 11 jezika i podržavaju ga različite platforme kao što su Windows, macOS i Linux operativni sustav. [8]



Slika 20. Kontrolna ploča XAMPP
Izvor: Autor

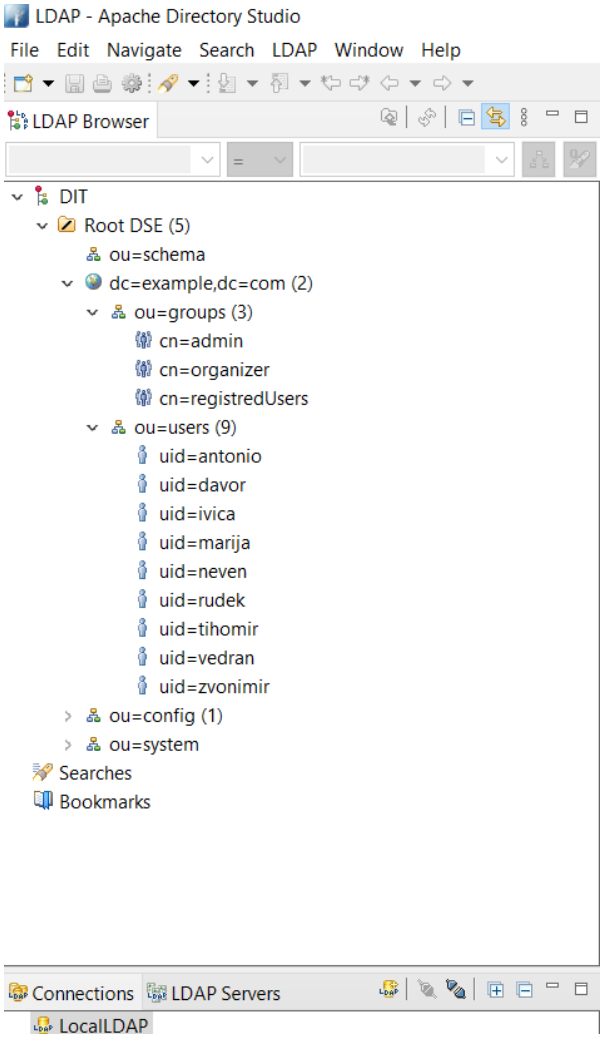
U projektu je platforma korištena kao poslužitelj MySQL baze podataka. Vrlo ga je jednostavno koristiti i vrlo je koristan prilikom razvoja, testiranja studentskih radova, ali i većih projekata u većim organizacijama prije samoga naseljavanja na glavni poslužitelj. Na slici 20. vidi se kontrolna ploča koja omogućava pokretanje željenih servisa.

3.3.3. Apache Directory Studio

Apache Directory studio je Eclipse bazirani LDAP preglednik koji se koristi s Apache DS LDAP poslužiteljem. Sastoji se od nekoliko dijelova [9]:

- LDAP Browsera – dizajniran je da radi s gotovo svakim LDAP poslužiteljem. Omogućuje čitanje i pregled strukture LDAP poslužitelja, ali i kreiranje, ažuriranje i brisanje zapisa.
- Schema Editor – omogućuje editiranje datoteka shema u OpenLDAP formatu.
- ApacheDS Configuration – koristi se za konfiguriranje Apache DS poslužitelja (server.xml i config.ldif datoteka).
- LDIF Editor – alat je za editiranje LDIF datoteka.

U sklopu projekta Apache Directory Studio koristio se za početno postavljanje uloga koje su se koristile u aplikaciji, a nakon toga za provjeru ispravnosti rada aplikacije kad je testirano dodavanje, ažuriranje i brisanje korisnika aplikacije.

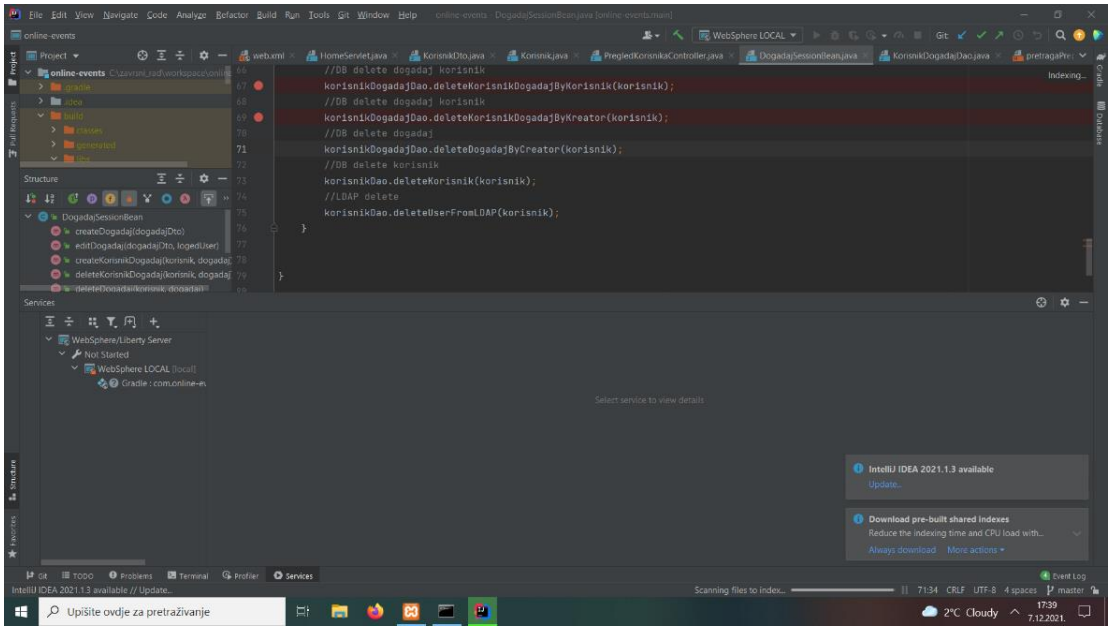


Slika 21. Apache Directory Studio LDAP Browser
Izvor: Autor

3.3.4. IntelliJ IDEA

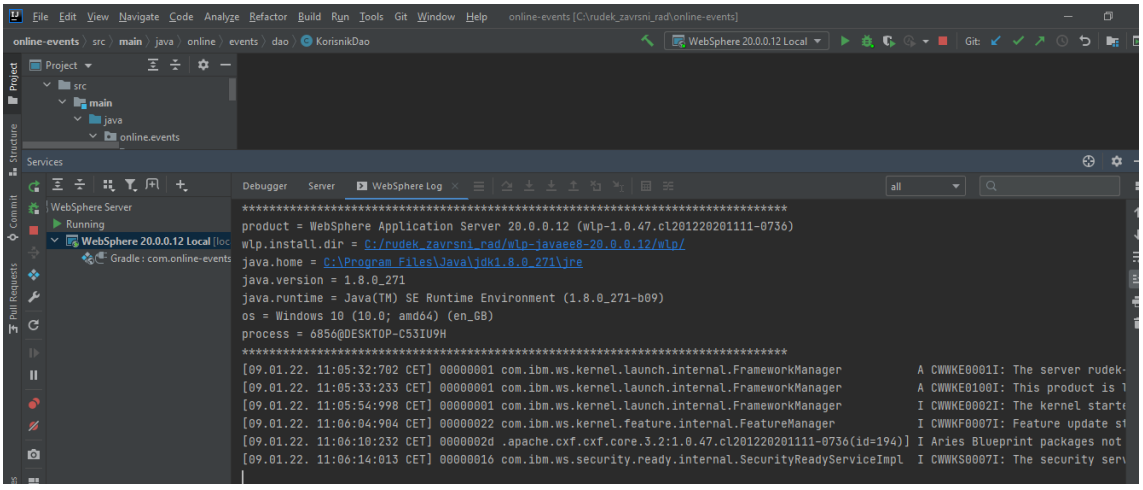
IntelliJ IDEA integrirano je razvojno okruženje za JVM (Java Virtual Machine) jezike. Radi rutinske i ponavljajuće zadatke pružajući pametno kompletiranje koda, statičku analizu koda.[10]

Sučelje IntelliJ-a prikazano je na slici 22., jednostavno je za korištenje i omogućava jednostavnu navigaciju kroz dijelove projekta.



Slika 22. IntelliJ IDEA
Izvor: Autor

U projektu je IntelliJ IDEA razvojno okruženje korišteno za pisanje koda u Java programskom jeziku, ponekad i za pisanje SQL uputa (u slučajevima kad se ne koristi MySQL Workbench). Također, IntelliJ IDEA omogućuje integraciju aplikacijskoga poslužitelja (u ovom slučaju IBM WebSphere Liberty aplikacijskoga poslužitelja) na vrlo jednostavan način čime je razvoj mnogo kraći i lakši. Na slici 23. prikazano je pokretanje aplikacijskoga poslužitelja iz IntelliJ-a.



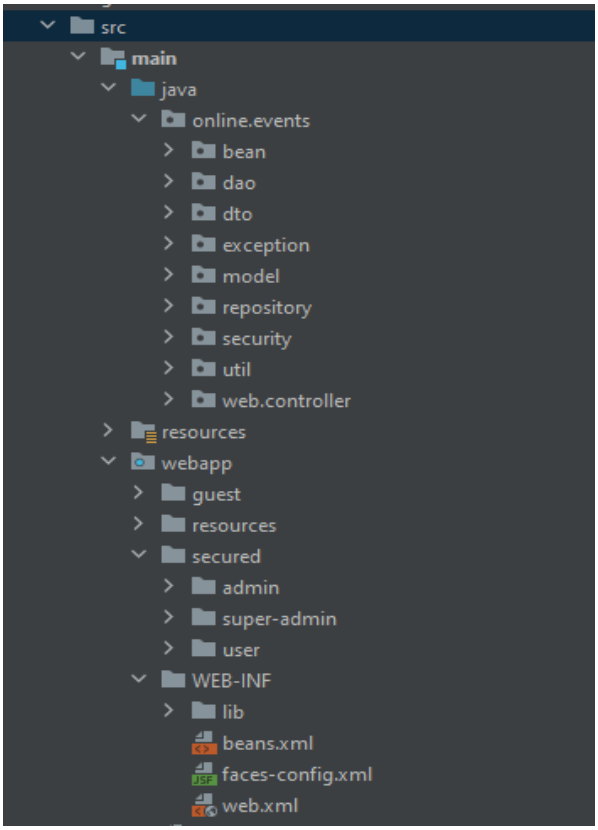
Slika 23. IntelliJ IDEA – Pokretanje aplikacijskoga poslužitelja
Izvor: Autor

4. Aplikacija

4.1. Struktura projekta

Struktura projekta zapravo je struktura ili raspored direktorija i pripadajućih datoteka. Većina je datoteka u ovome projektu Java klase, a osim njih imamo i xhtml datoteke, slike i xml datoteke. Datoteke su raspoređene u direktorije koje u Java projektima nazivamo Java paketi. Java paketi na disku zapravo i jesu direktoriji. Sve Java klase ili datoteke koje ubrajamo u isti paket zapravo su smještene u istome direktoriju.

Slika 24. prikazuje strukturu projekta aplikacije za prezentiranje društvenih događaja. Kao i u svakom Java projektu na početku imamo korijenski direktorij unutar kojega su poddirektoriji za svaki paket.



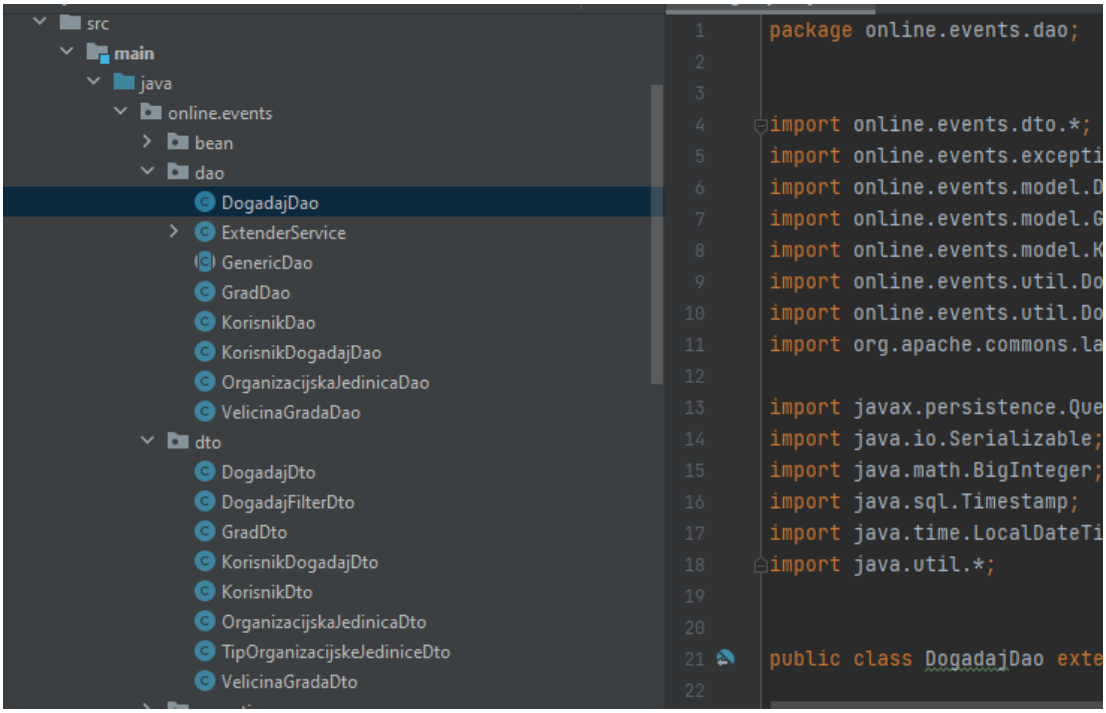
Slika 24. Struktura projekta

Izvor: Autor

Unutar korijenskoga direktorija postoji main direktorij, a unutar njega direktoriji:

- java – Sadrži sve Java klase (izvorni kod) koje se koriste u projektu i može se smatrati najvažnijim dijelom projekta.
- resources – Uobičajeno je mjesto na koje se spremaju konfiguracijske datoteke, uglavnom u xml formatu.
- webapp – Sadrži resurse kao što su html datoteke, xhtml datoteke, css datoteke.

Na sljedećoj slici prikazan je online.event.dao paket koji se nalazi unutar main/java direktorija. Unutar njega smještene su Java klase koje koristimo u sloju pristupa podatcima, npr. DogadajDao klasa, KorisnikDao klasa.



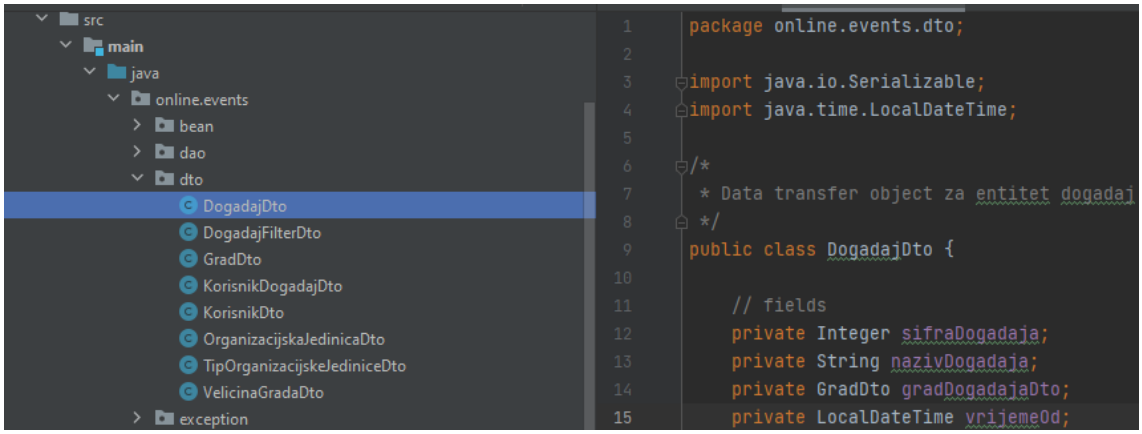
Slika 25. online.events.dao paket

Izvor: Autor

Online.events.dto paket (slika 25.) sadrži Java klase koje služe, odnosno koje se koriste kao objekti transfera podataka između svih triju slojeva aplikacije. Neki od primjera su DogadajDto klasa, KorisnikDto klasa.

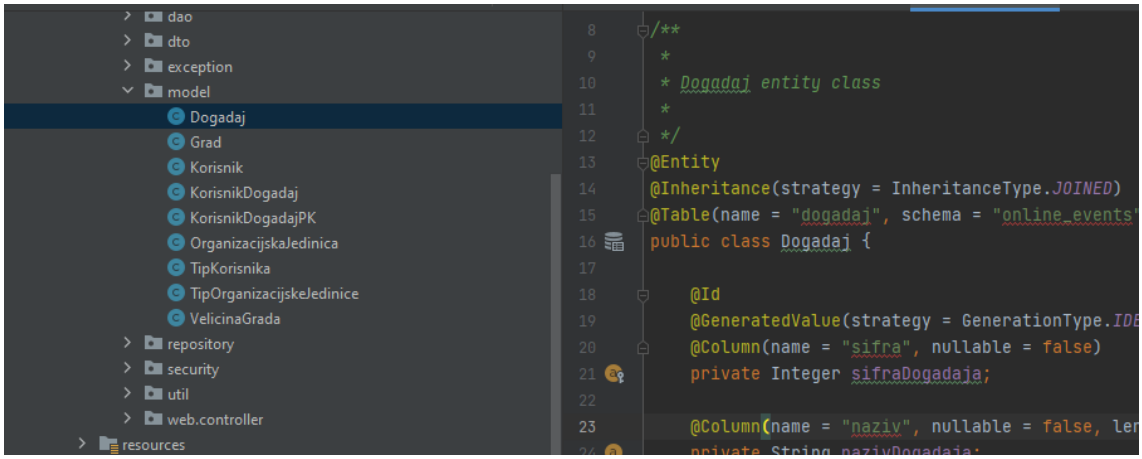
Online.events.model paket (slika 26.) sadrži Java klase koje predstavljaju JPA (Java Persistence API) entitete opisane u poglavlju sloja pristupa podatcima. Neki od primjera su Dogadaj klasa, Korisnik klasa.

Online.events.model (slika 27.) sadrži pakete koji se odnose na klasu Dogadaj.



Slika 26. online.events.dto paket

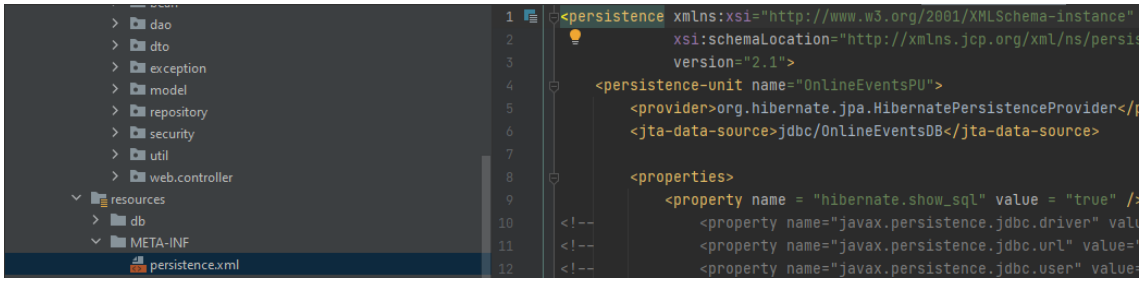
Izvor: Autor



Slika 27. online.events.model paket

Izvor: Autor

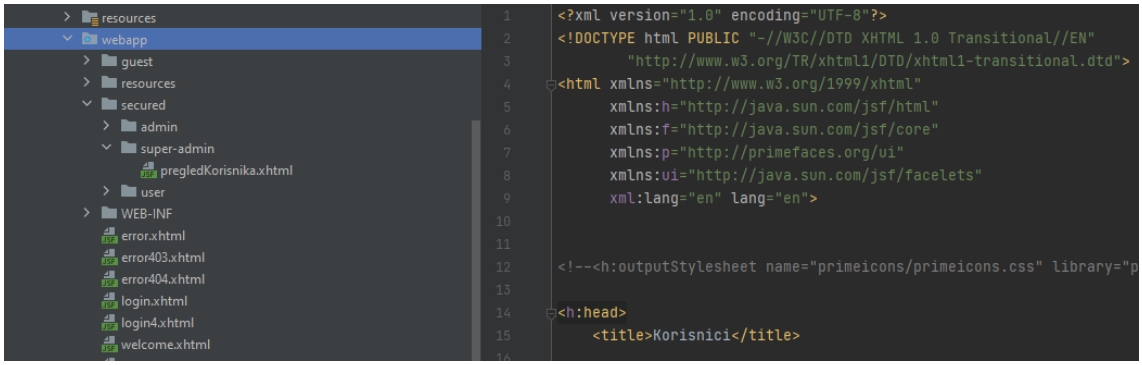
Kao što je već spomenuto, resources direktorij uobičajeno je mjesto kamo se spremaju konfiguracijske datoteke, uglavnom u xml formatu. Na slici 28. primjer je konfiguracijske datoteke persistence.xml u kojoj je definiran izvor baze podataka, odnosno naziv izvora baze podataka (jdbc/OnlineEventsDB) čije su postavke definirane u konfiguraciji aplikacijskoga poslužitelja.



Slika 28. resource direktorij

Izvor: Autor

Zadnji je direktorij webapp direktorij koji sadrži datoteke prezentacijskoga sloja aplikacije xhtml datoteke, css datoteke i ostale resurse potrebne za korisničko sučelje, što je prikazano na slici 29.

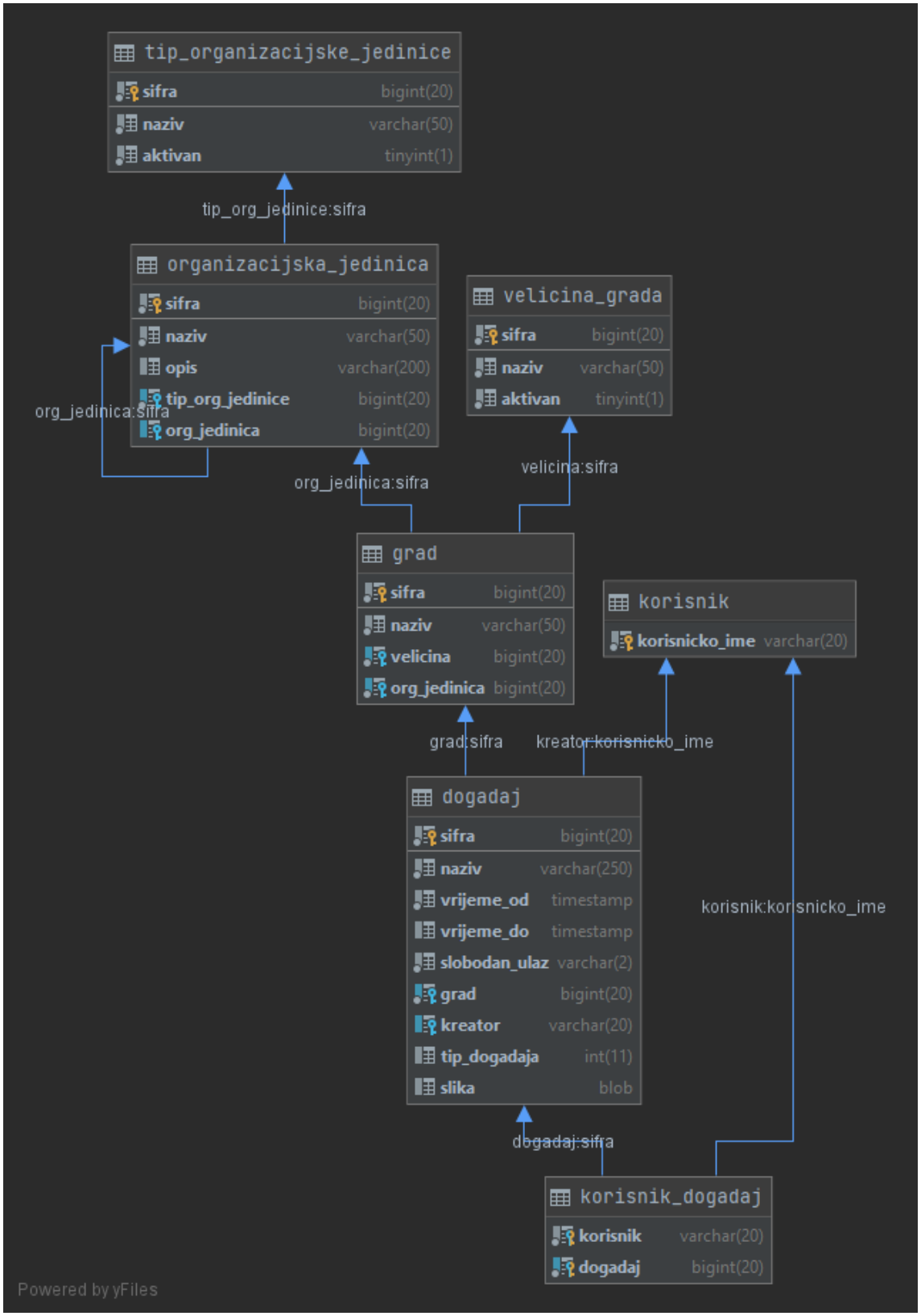


Slika 29. webapp direktorij

Izvor: Autor

4.2. Model baze podataka

Model baze podataka prikazuje logičku strukturu baze podataka, uključujući odnose i ograničenja koja određuju kako se podatci mogu pohraniti i kako se njima može pristupiti. Pojedinačni modeli baze podataka dizajnirani su na temelju pravila i koncepata onoga šireg modela podataka koji dizajneri usvoje. Većina modela podataka može se predstaviti popratnim dijagramom baze podataka. Postoje različiti modeli baze podataka, a neki poznatiji su hijerarhijski model baze podataka, relacijski model, mrežni model, zvjezdasti model te objektno orijentirani model. Većina sustava za upravljanje bazama podataka izgrađena je s posebnim modelom podataka i zahtijeva od korisnika da usvoje željeni model, ali neki modeli mogu sadržavati više modela. Na slici 30. prikazan je model baze podataka aplikacije. Sastoji se od sljedećih tablica:

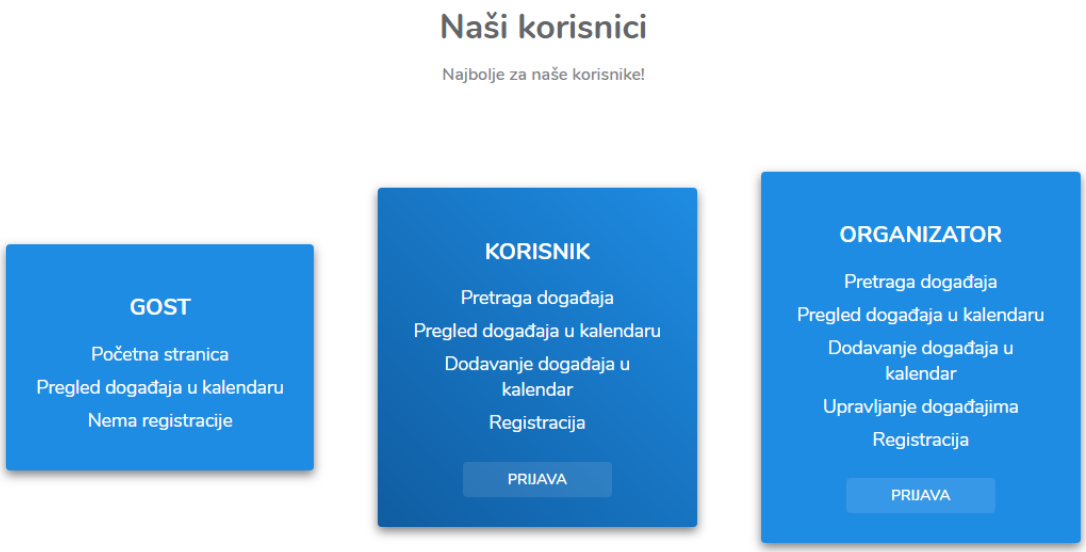


Slika 30. Model baze podataka

Izvor: Autor

4.3. Uloge i korisnici aplikacije

U aplikaciji postoje četiri vrste korisnika, odnosno četiri korisničke uloge ili vrste korisnika, a to su administrator, organizator, registrirani korisnik te gost. Glavne funkcionalnosti, odnosno mogućnosti koje pojedina vrsta korisnika ima, vidljive su na početnoj stranici aplikacije (slika 31).

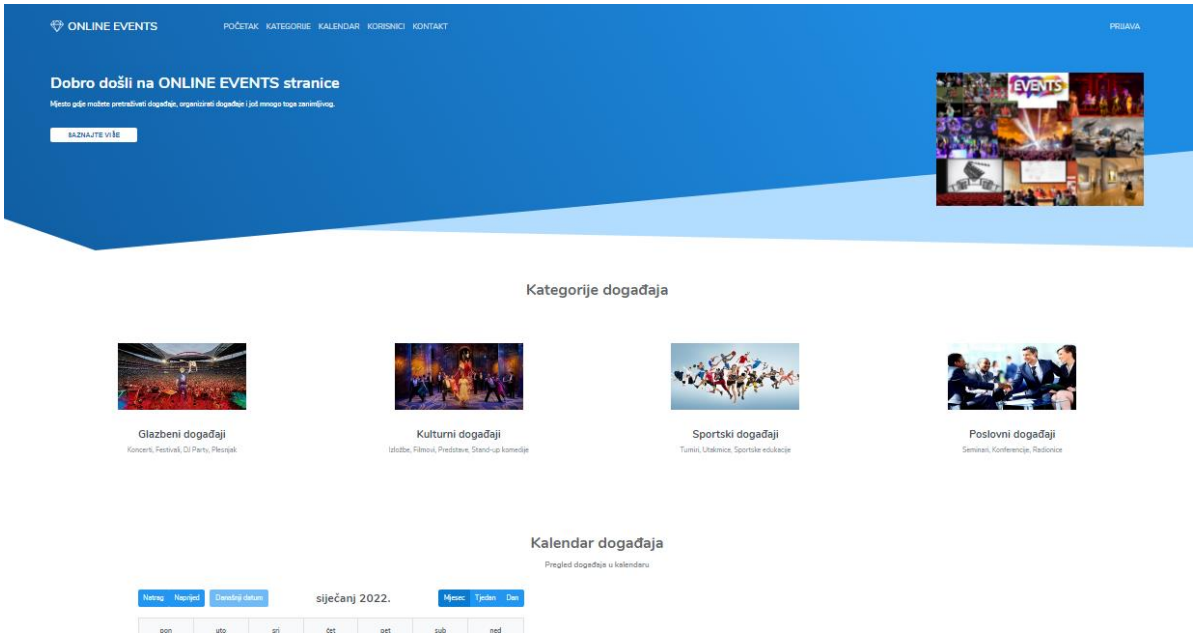


Slika 31. Dio početne stranice aplikacije s pregledom korisnika i funkcionalnostima

Izvor: Autor

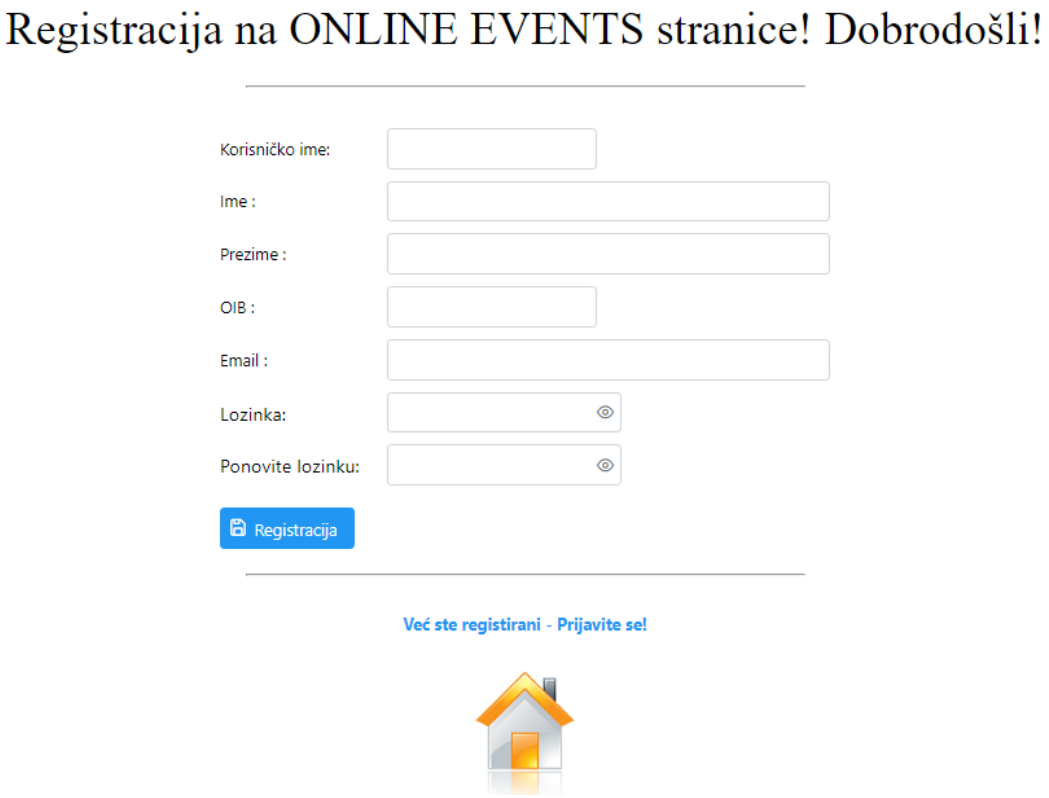
4.3.1 Gost

Svi korisnici aplikacije koji nisu registrirani ubrajaju se u grupu korisnika koju nazivamo gost. Oni imaju mogućnost pristupa početnoj stranici i pregleda početne stranice aplikacije na kojoj se nalazi pregled glavnih funkcionalnosti, vrsta korisnika, osnovni kalendar događaja (slika 31). Cilj je pregleda početne stranice da privuče korisnike koji nisu registrirani da se odluče za neku od uloga koje omogućuju naprednije funkcionalnosti ovisno o potrebama korisnika. Svakom posjetitelju početne stranice aplikacije, odnosno gostu, omogućena je registracija (slika 33) kojom zapravo postaje registrirani korisnik aplikacije. Na slici 32. prikazana je početna stranica.



Slika 32. Dio početne stranice aplikacije s pregledom kategorija događaja i kalendara događaja

Izvor: Autor

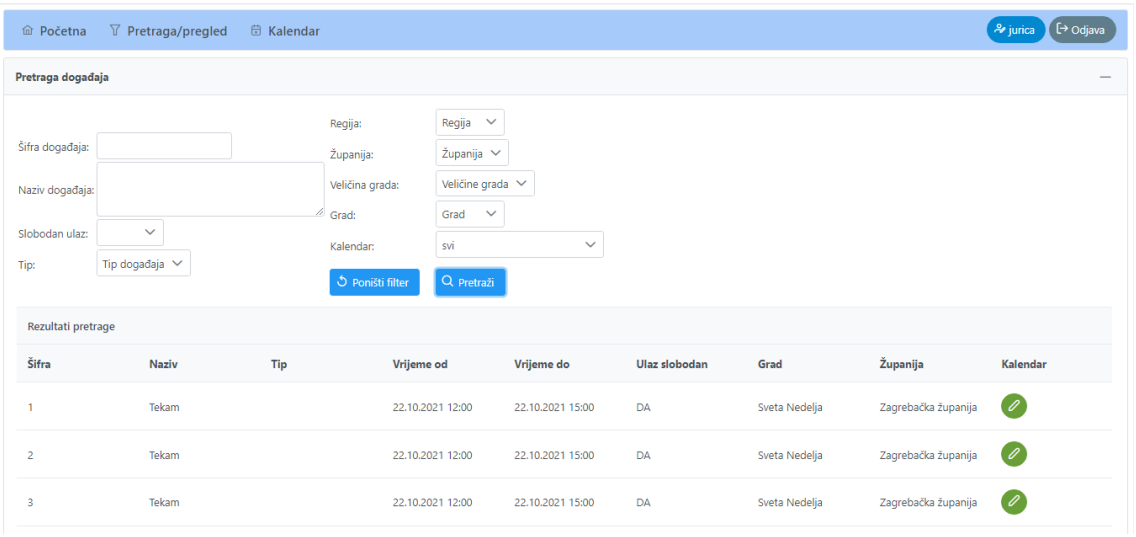


Slika 33. Stranica za registraciju korisnika u aplikaciju

Izvor: Autor

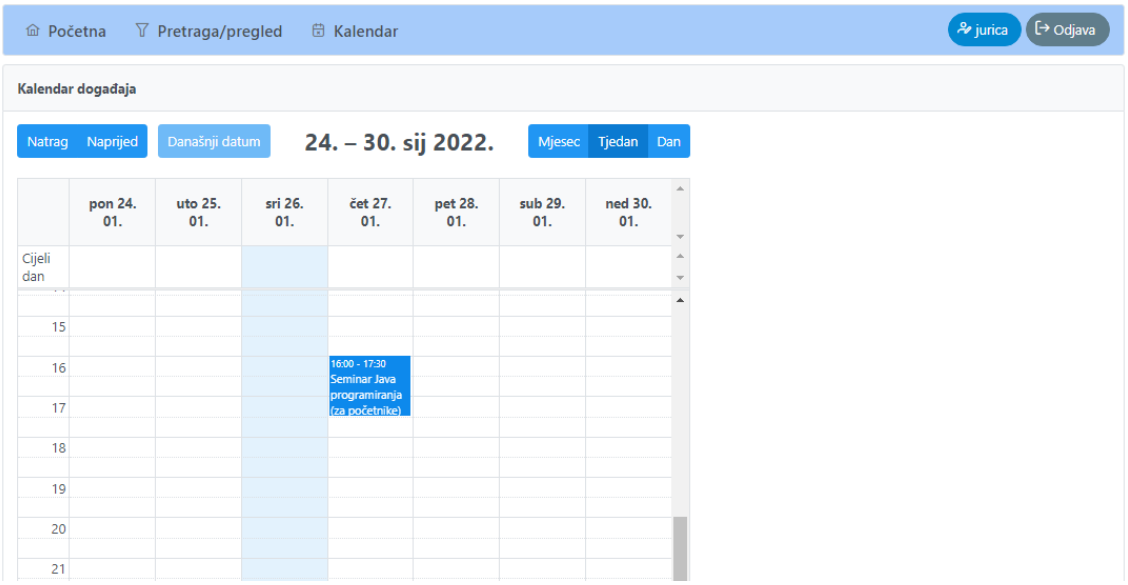
4.3.2 Registrirani korisnik

Registrirani korisnik druga je uloga u aplikaciji, a u odnosu na korisnike koji se ubrajaju u goste registrirani korisnici imaju i mogućnost pretrage i pregleda događaja, dodavanje i uklanjanje događaja u kalendar i iz kalendara, pregled događaja u kalendaru. Na slikama 34. i 35. može se vidjeti koje mogućnosti imaju registrirani korisnici. U kalendaru se događaji mogu pretraživati po danu, tjednu i mjesecu, kao što je prikazano na slikama 35., 36. i 37.



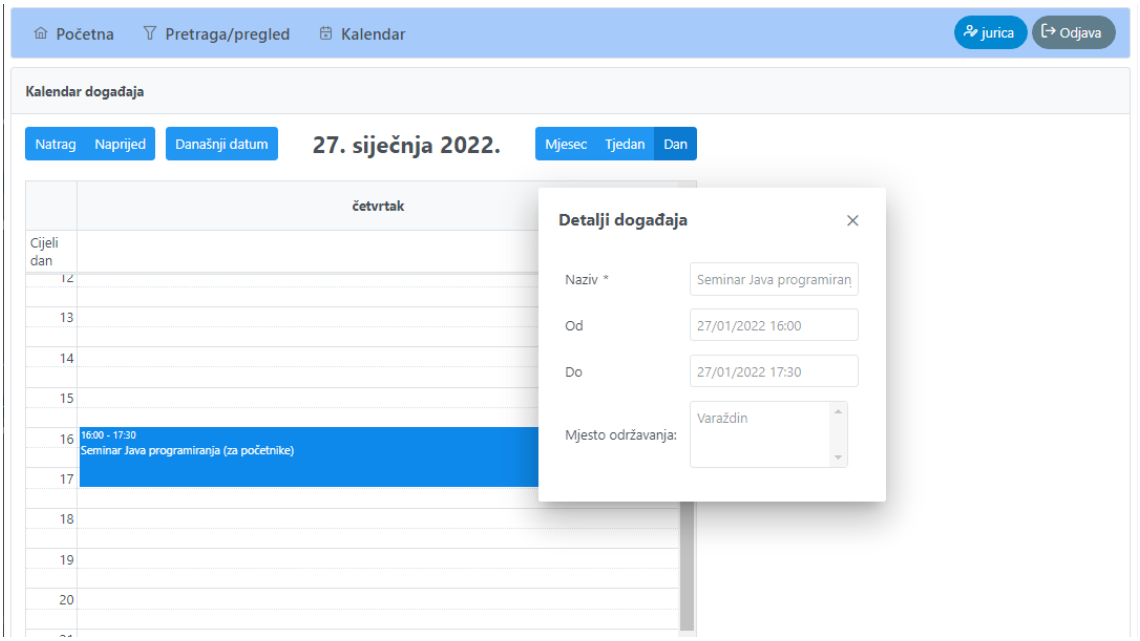
Slika 34. Stranica pretrage i pregleda događaja

Izvor: Autor



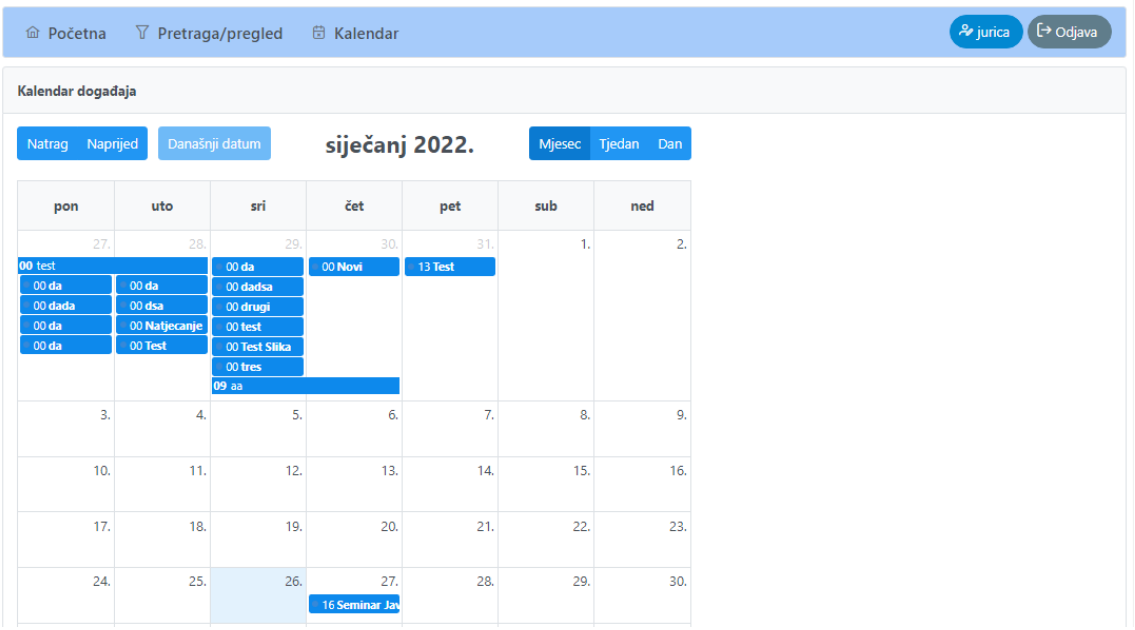
Slika 35. Stranica kalendara – tjedni pregled

Izvor: Autor



Slika 36. Stranica kalendara – dnevni pregled

Izvor: Autor

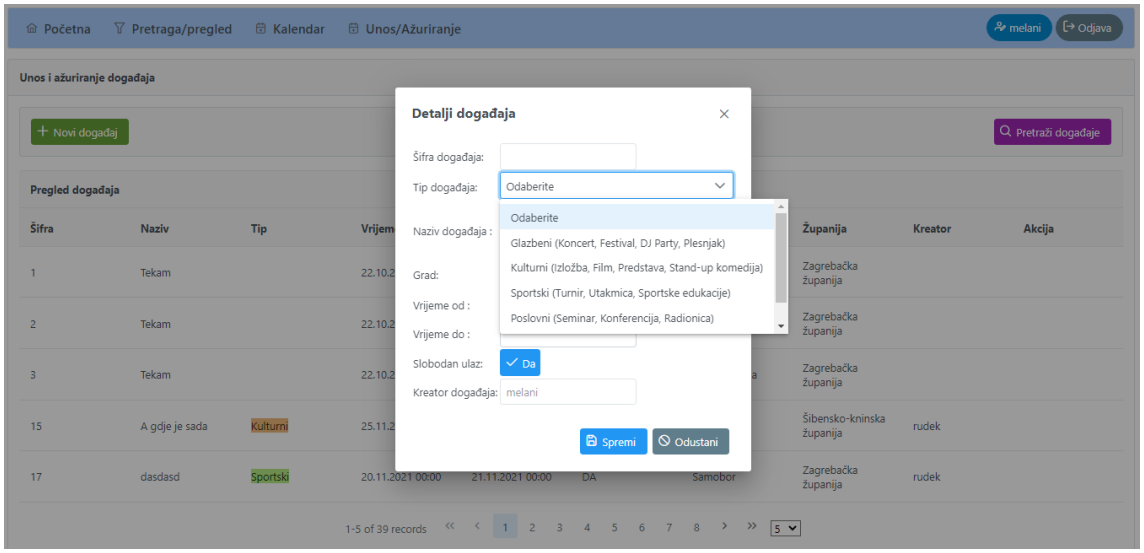


Slika 37. Stranica kalendara – mjesečni pregled

Izvor: Autor

4.3.3 Organizator

Organizatori su korisnici koji u odnosu na registrirane korisnike i korisnike koji se ubrajaju u goste imaju i mogućnost upravljanja događajima, odnosno mogu dodavati, ažurirati i brisati događaje. Ažuriranje i brisanje događaja omogućeno im je samo za događaje koje su oni unijeli, odnosno kreirali.

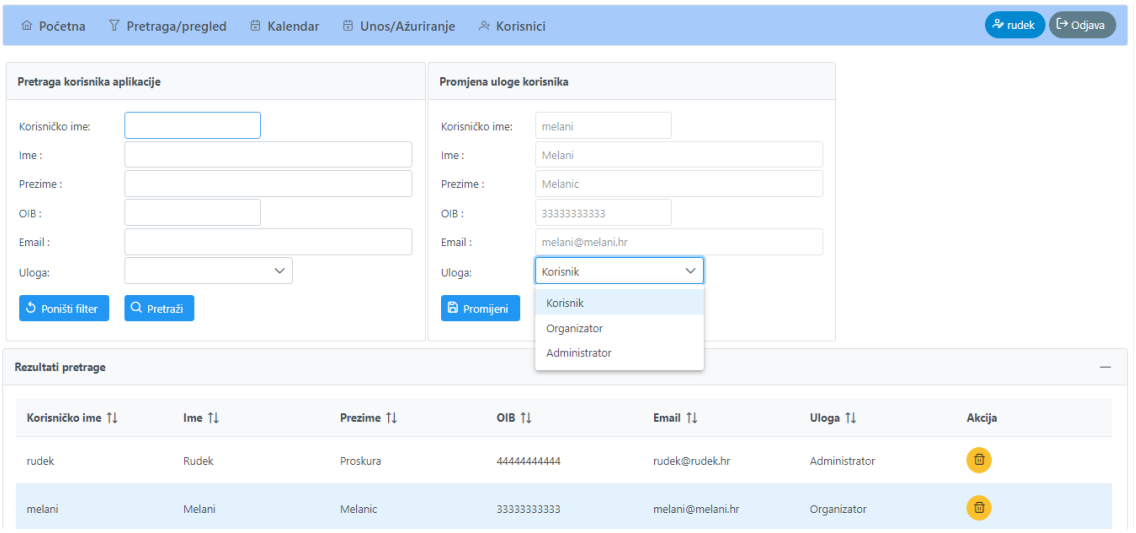


Slika 38. Unos/ažuriranje događaja

Izvor: Autor

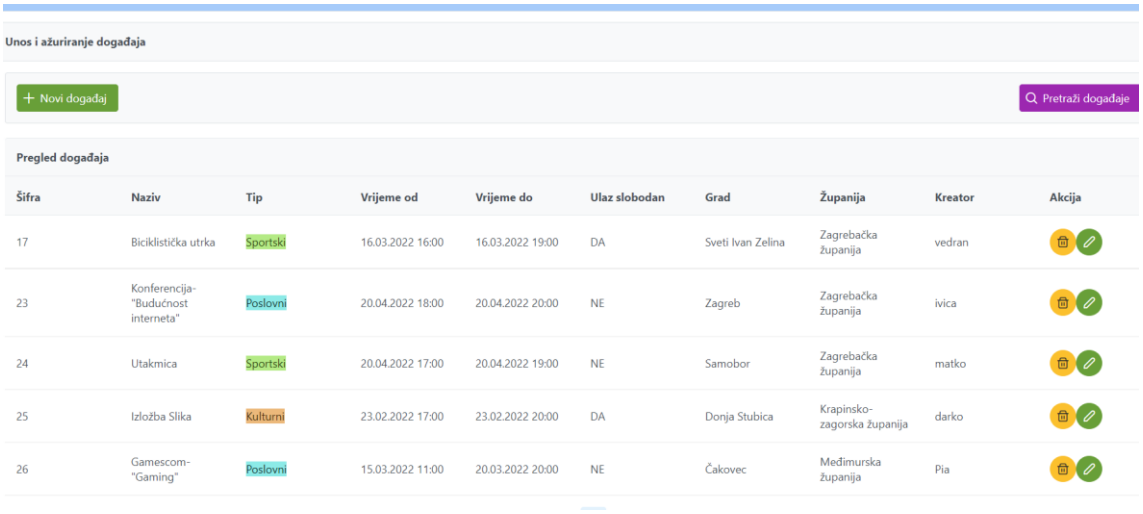
4.3.3 Administrator

Administratori imaju najviše mogućnosti i prava u aplikaciji. Osim dosad spomenutih funkcionalnosti i mogućnosti mogu upravljati korisnicima, odnosno, mogu pretraživati korisnike te im mijenjati korisničke uloge. Također, administrator ima i ovlasti da može brisati ili ažurirati neželjeni sadržaj, odnosno objave organizatora. Slika 39. prikazuje kako administrator može upravljati korisnicima, mijenjati uloge te obrisati pojedine korisnike. Na slici 40. prikazano je da administrator može ukloniti ili ažurirati pojedini događaj.



Slika 39. Stranica upravljanja korisnika

Izvor: Autor



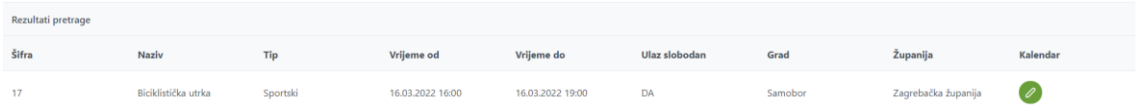
Slika 40. Stranica upravljanja administratora

Izvor: Autor

4.4. Opcije i mogućnosti aplikacije

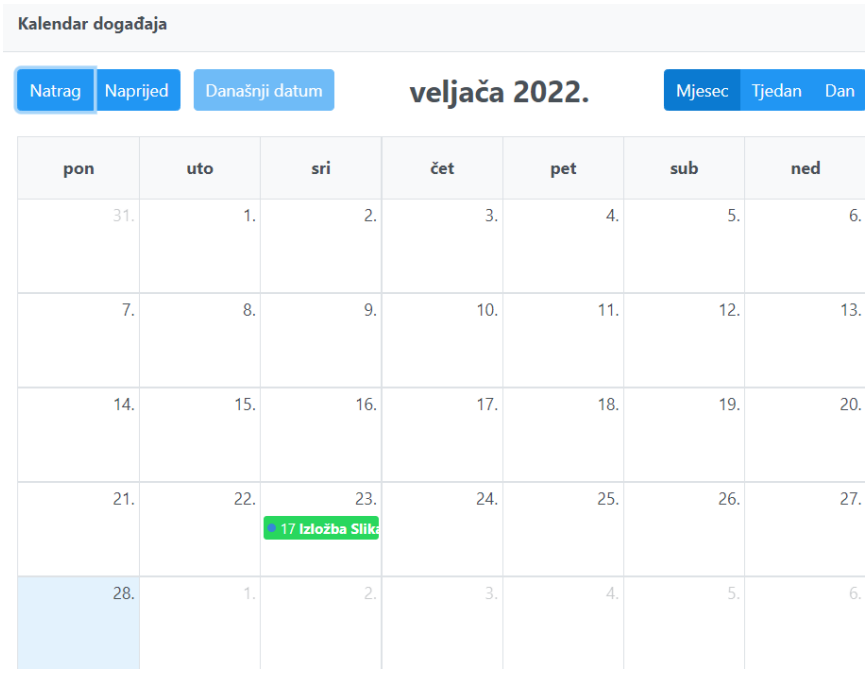
Dodavanje događaja

Dakle, dodavati događaje u kalendar mogu registrirani korisnici, organizatori i administratori koji su ujedno i organizatori. Da se doda određeni događaj, najprije se pretražuju događaji. Klikom na gumb Pretraži ponude se svi događaji. Kad se odabere događaj koji zanima korisnika, klikne se na zeleni gumb i događaj se dodaje u korisnikov kalendar (slika 41). Nakon što je korisnik odabrao događaj, odabrani se događaj pojavi u kalendaru i obojan je zelenom bojom (slika 42).



Slika 41. Dodavanje događaja

Izvor: Autor

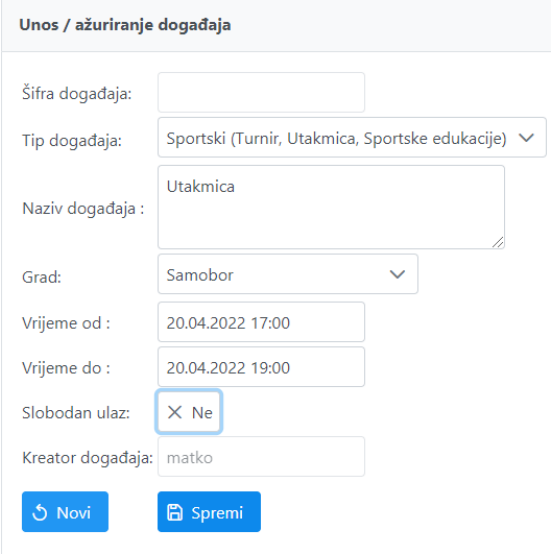


Slika 42. Događaj u kalendaru

Izvor: Autor

Kreiranje događaja

Kad se kreira događaj, najprije se odabere tip događaja, odredi se je li to glazbeni, kulturni, sportski ili poslovni događaj. Nakon što se odabere tip događaja, daje se naziv događaju, odabere se grad, vrijeme trajanja događaja, odredi se je li ulaz slobodan ili nije. Kad se to napravi, klikom na gumb Spremi pohrani se sve uneseno. Postupak kreiranja događaja prikazan je na slici 43.

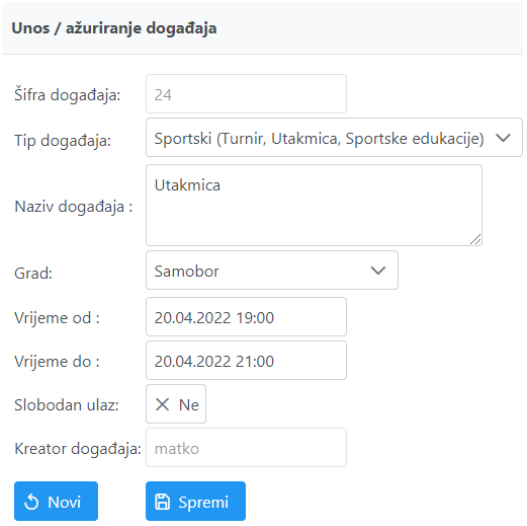


Slika 43. Kreiranje događaja

Izvor: Autor

Ažuriranje događaja

Kod ažuriranja događaja kreator odabere događaj koji je kreirao, zatim odabere stavke koje želi promijeniti te ih sprema. Na slici 43. vidi se promjena vremena događaja koji je prikazan na slici 44.

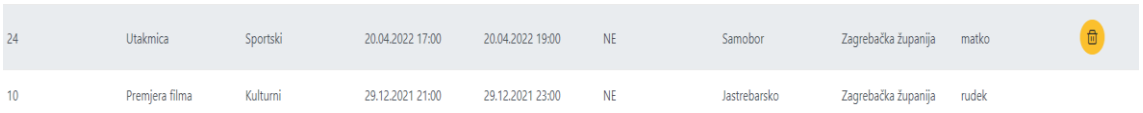


Slika 44. Ažuriranje događaja

Izvor: Autor

Brisanje događaja

Određeni događaj može izbrisati samo autor toga događaja. Da bi se događaj izbrisao, autor mora kliknuti na žuti gumb koji označava brisanje. Na slici 45. prikazan je način brisanja događaja. Jednom kad se događaj obriše, on više nije vidljiv na popisu događaja.



Slika 45. Brisanje događaja

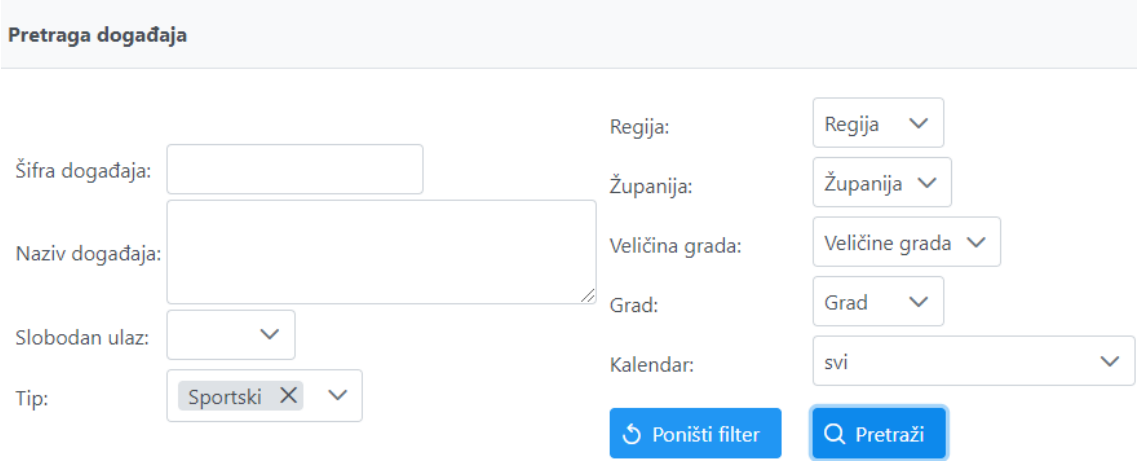
Izvor: Autor

Pretraživanje događaja

Događaji se mogu pretraživati na nekoliko načina.

Prvi je način da se klikne na gumb Pretraži nakon čega se prikažu svi događaji.

Drugi je način pretraživanja događaja da se oni pretražuju po ponuđenim atributima kao što su šifra događaja, naziv, po tome je li ulaz slobodan ili ne, po tipu događaja, po regiji, županiji, veličini grada, po nazivu grada. Pretraživanje događaja prikazano je na slici 46.

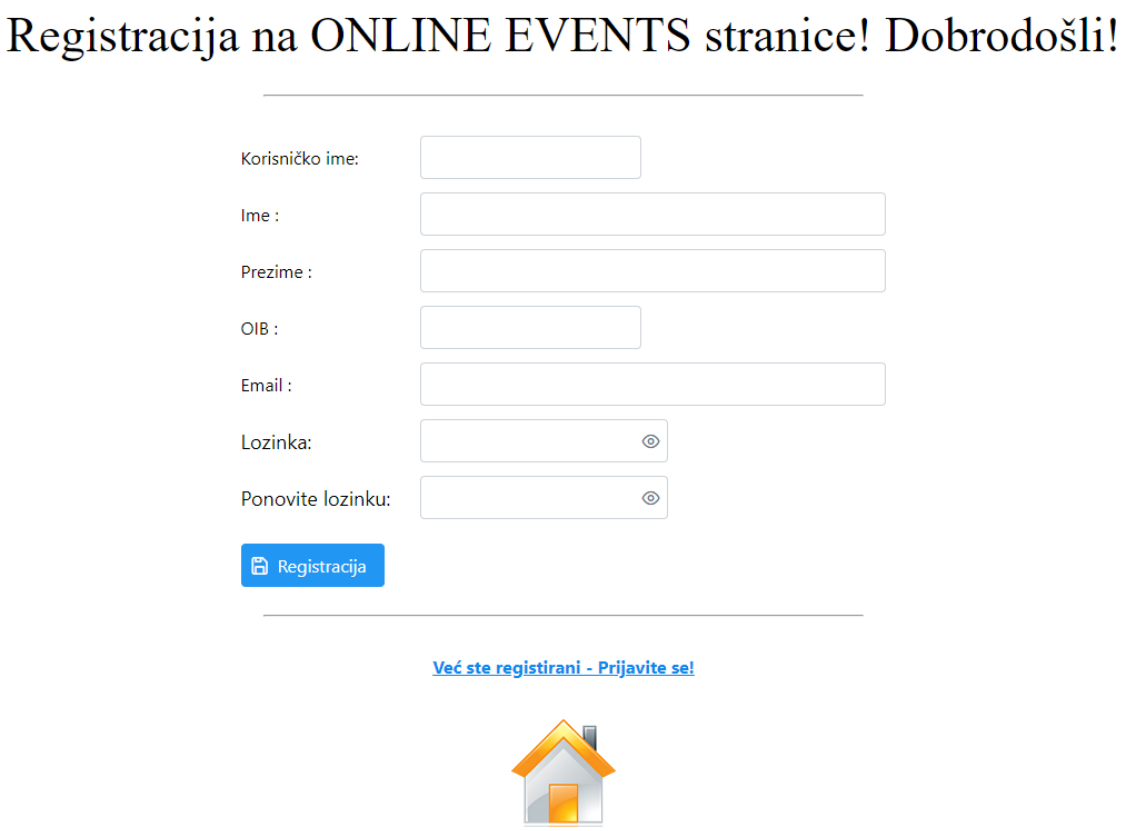


Slika 46. Pretraživanje događaja

Izvor: Autor

Dodavanje korisnika

Dodavanje korisnika, odnosno registracija korisnika obavlja se putem forme za registraciju korisnika, što je prikazano na slici 47. Da bi se korisnik mogao registrirati, mora ispuniti formular u koji mora upisati svoje ime, prezime, OIB, adresu e-pošte te si odabire lozinku kojom se prijavljuje i korisničko ime. U slučaju da neki podatci kod unosa nisu ispravni, javlja se poruka o pogrešci.

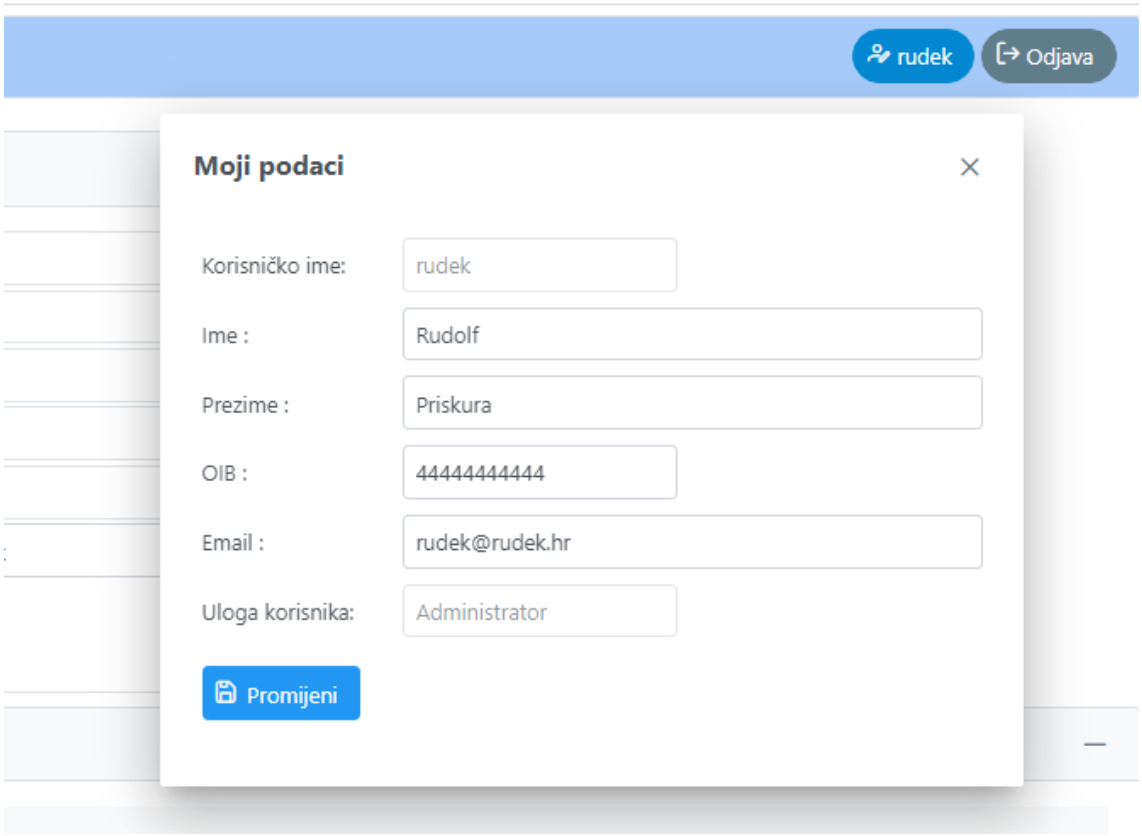


Slika 47. Dodavanje korisnika

Izvor: Autor

Uređivanje korisnika

Korisnik na svojemu profilu može promijeniti ime, prezime, OIB i adresu e-pošte. Ulogu korisnika može promijeniti samo administrator. Na slici 48. prikazano je kako korisnik može mijenjati, odnosno uređivati svoje podatke.



Slika 48. Uređivanje korisnika

Izvor: Autor

Brisanje korisnika

Korisnika može izbrisati samo administrator klikom na žuti gumb za brisanje (slika 49). Kad se korisnik izbriše, nema ga više u listi korisnika, niti se više taj korisnik može ulogirati na stranicu.

Slika 49. Brisanje korisnika

Izvor: Autor

5. Zaključak

U današnje vrijeme kad se mnogo toga temelji na računalima i na virtualnome svijetu koji je prisutan u svim sferama života, od škola do poslovanja u velikim organizacijama, raste i broj različitih web-aplikacija.

Pri izradi ovoga rada korištene su Java tehnologije, kao što su CSS, HTML, JSF i JavaScript. Navedene su tehnologije vrlo jednostavne za korištenje jer postoji mnogo dokumentacije koja olakšava rad s tim tehnologijama, ali je isto tako bitan dio uspješnosti korištenja i predznanje koje se stječe školovanjem.

6. Popis literature

[1]Java, https://www.ibm.com/docs/en/was-liberty/zos?topic=overview-jakarta-java- ee-8-in-liberty

[2] Arhitektura, https://www.ibm.com/cloud/learn/three-tier-architecture

[3] Prednosti troslojne arhitekture,https://www.geeksforgeeks.org/advantages-and-disadvantages-of-three-tier-architecture-in-dbms/

[4]Session Bean, https://www.developer.com/java/enterprise-java/introducing-jee-application-session-beans/

[5]Apache Directory LDAP API , https://jumpcloud.com/blog/what-is-ldap

[6]Apache Directory, https://directory.apache.org/apacheds/basic-ug/1.1-what-apacheds-is.html

[7]MySql, https://www.mysql.com/products/workbench/

[8]Xampp, https://www.javatpoint.com/xampp

[9]Apache Directory Studio, https://directory.apache.org/studio/

[10]IntelliJ IDEA, https://www.jetbrains.com/help/idea/discover-intellij-idea.html

Prilozi

Popis slika

Slika 1. Troslojna arhitektura

Slika 2. Arhitektura i korištene tehnologije

Slika 3. Komponente MVC uzorka dizajna

Slika 4. Dio korisničkoga sučelja/ekrana na opciji Unos / Ažuriranje događaja

Slika 5. Vrste Session Beana

Slika 6. Primjer Stateless Session Beana

Slika 7. Primjer validacije podataka za događaj prije samog spremanja u bazu podataka

Slika 8. Primjer validacije podataka za korisnika prije samog spremanja u bazu podataka

Slika 9. Primjer redoslijeda akcije prigodom brisanja korisnika

Slika 10. Veza između sloja poslovne logike i baze podataka (ORM)

Slika 11. JPA entitet Dogadaj

Slika 12. Primjer korištenja EntityManager metoda u projektu

Slika 13. Grupe LDAP direktorija

Slika 14. Pregled atributa LDAP korisnika

Slika 15. Korisnici LDAP grupe registredUsers

Slika 16. Dodavanje novoga korisnika

Slika 17. Pretraga korisnika prema korisničkom imenu i punjenje objekta KorisnikDto s podatcima korisnika

Slika 18. Promjena adrese e-pošte korisnika

Slika 19. Premještanje korisnika iz jedne u drugu grupu – promjena tipa korisnika

Slika 20. Kontrolna ploča XAMPP

Slika 21. Apache Directory Studio LDAP Browser

Slika 22. IntelliJ IDEA

Slika 23. IntelliJ IDEA – Pokretanje aplikacijskoga poslužitelja

Slika 24. Struktura projekta

Slika 25. online.events.dao paket

Slika 26. online.events.model paket

Slika 27. online.events.model paket

Slika 28. resource direktorij

Slika 29. webapp direktorij

Slika 30. Model baze podataka

Slika 31. Dio početne stranice aplikacije s pregledom korisnika i funkcionalnosti

Slika 32. Dio početne stranice aplikacije s pregledom kategorija događaja i kalendara događaja

Slika 33. Stranica za registraciju korisnika u aplikaciju

Slika 34. Stranica pretrage i pregleda događaja

Slika 35. Stranica kalendara – tjedni pregled

Slika 36. Stranica kalendara – dnevni pregled

Slika 37. Stranica kalendara – mjesečni pregled

Slika 38. Unos/ažuriranje događaja

Slika 39. Stranica upravljanja korisnika

Slika 40. Stranica upravljanja administratora

Slika 41. Dodavanje događaja

Slika 42. Događaj u kalendaru

Slika 43. Kreiranje događaja

Slika 44. Ažuriranje događaja

Slika 45. Brisanje događaja

Slika 46. Pretraživanje događaja

Slika 47. Dodavanje korisnika

Slika 48. Uređivanje korisnika

Slika 49. Brisanje korisnika