



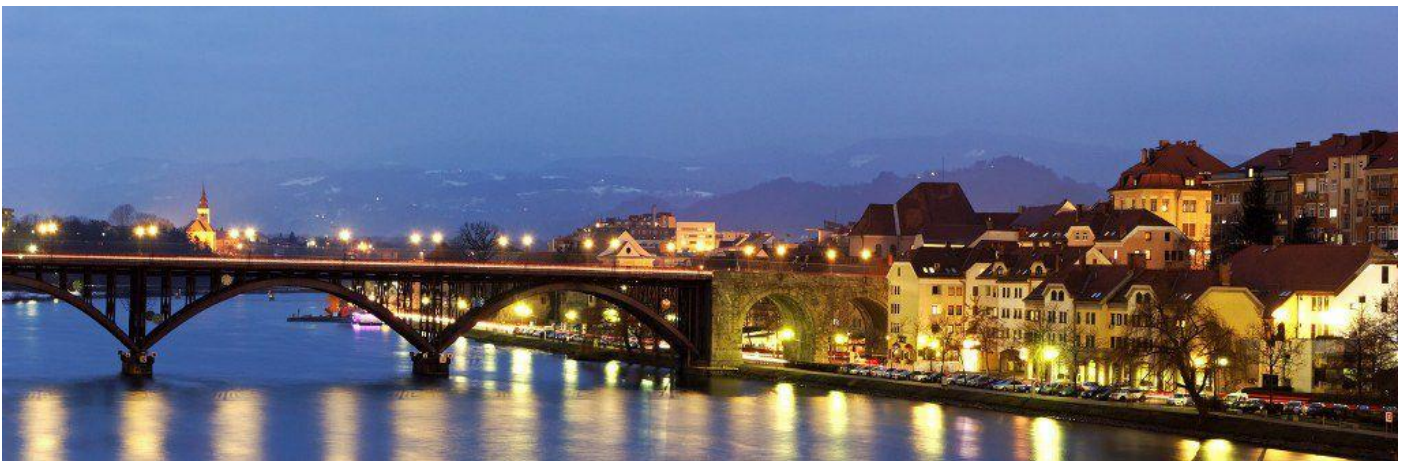
Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Fakulteta za elektrotehniko, računalništvo in informatiko

PRAKTIKUM I
1. LETNIK UNI (ITK)

MESTNI UTRIP



Avtorji:

Primož Stopar
Uroš Zagoranski
Aljoša Sikošek
Nejc Vnuk

Mentor:

Mitja Gradišnik

Junij 2018

VSEBINA

NAVODILA NALOGE	3
ENTITETNO RELACIJSKI MODEL	4
OPIS REŠITVE PROBLEMA	5
GLAVNE KOMPONENTE, METODE	7
1. INDEX.JSP (GLAVNA STRAN) :	7
2. REGISTRACIJA :	7
3. STRANI, DOSTOPNE PREKO FILTROV NA GLAVNI STRANI :	7
4. PODROBNOSTI :	8
5. DODAJANJE DOGODKA :	8
6. MOJI DOGODKI :	8
7. UREJANJE DOGODKOV :	8
UPORABNIŠKI VMESNIK APLIKACIJE	9
POMEMBNEJŠI IZSEKI KODE	13
POROČILO TESTIRANJA	16

KAZALO SLIK

SLIKA 1: NAVODILA NALOGE	3
SLIKA 2: ENTITETNO RELACIJSKI MODEL	4
SLIKA 3: ODSLOTKI JEZIKOV	5
SLIKA 4: LENT	6
SLIKA 5: POZDRAVNA STRAN INDEX.JSP	9
SLIKA 6: GALERIJA FILTRIRANIH SEZNAMOV	9
SLIKA 7: DODAJANJE DOGODKA	10
SLIKA 8: UREJANJE ŽELJENEGA DOGODKA	10
SLIKA 9: AKTUALNI DOGODKI	11
SLIKA 10: PODROBNOSTI IZBRANEGA DOGODKA	11
SLIKA 11: GALERIJA SLIK PRI PODROBNOSTIH	12
SLIKA 12: ZEMLJEVID PRI PODROBNOSTIH	12
SLIKA 13: KOMENTARJI	13
SLIKA 14: CONTROLLER DODAJANJE DOGODKA	13
SLIKA 15: SORTIRANO BRANJE DOGODKA IZ BAZE	14
SLIKA 16: JQUERY PRI REGISTRACIJI	14
SLIKA 17: JSTL	15
SLIKA 18: JAVANSKI RAZRED	15
SLIKA 19: TESTIRANJE	16

NAVODILA NALOGE

Mestni utrip

Mentor: Mitja Gradišnik

Načrtujte in izdelajte sistem namenjen spletni skupnosti, ki jo zanima aktualno dogajanje v mestu. Vaš sistem naj torej cilja na turiste, ki obišejo mesto, kot tudi lokalnim prebivalcem.

Uporabnikom spletne skupnosti omogočite dodajanje restavracij, klubov, znamenitosti ter drugih prizorišč dogajanja v mestu. Uporabniki se morajo deliti v dve skupini. V prvo skupino sodijo uporabniki, ki objavljajo in soustvarjajo dogodke v mestu, na drugi strani pa uporabniki, ki želijo biti obveščeni o dogajanju v mestu.

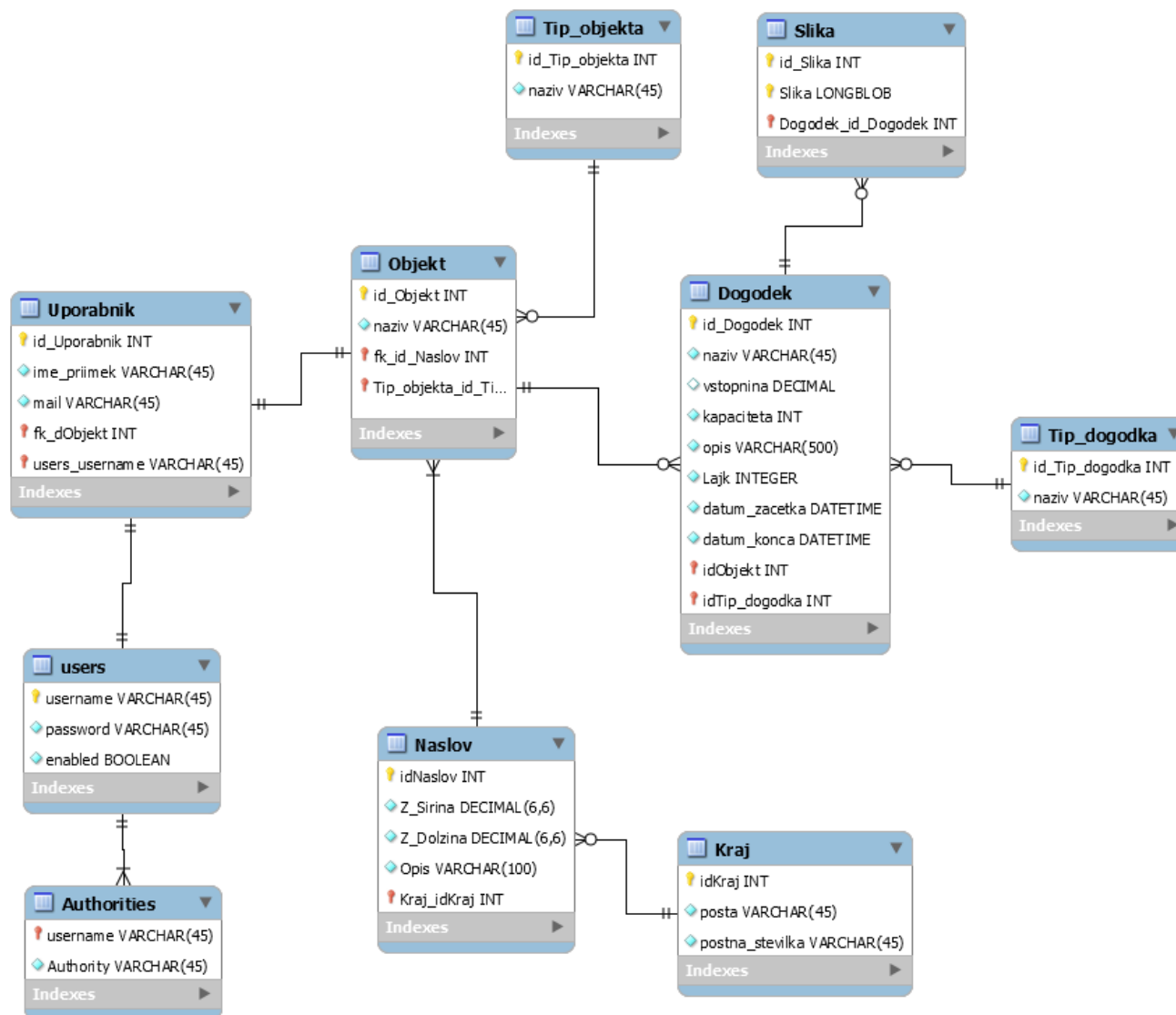
Da bi lahko uporabniki portala lažje našli za jih najprimernejše dogodke, implementirajte tudi napredno iskanje po aktualnih dogodkih.

Dogodki naj bodo uporabnikom prikazani na dva načina:

- glede na datum,
- glede na prizorišče.

Uporabniki spletnega portala naj imajo možnost preklapljanja med tema dvema pogledoma.

ENTITETNO RELACIJSKI MODEL

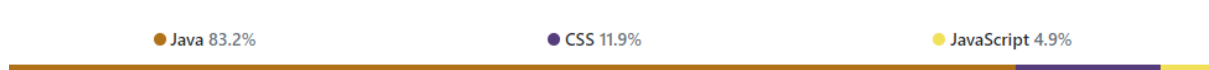


SLIKA 2: ENTITETNO RELACIJSKI MODEL

OPIS REŠITVE PROBLEMA

Aplikacija je zasnovana tako, da je uporabniku kar se da prijazna. Na zgornjem delu ekrana je konsistentno oblikovan meni, v katerem se nahajajo pomembnejše komponente. Celotna stran je oblikovana z dodatkom Bootstrap-a, zaradi česar je prilagodljiva za vse naprave. Osnova aplikacije je podatkovna baza v MySQL-u, celotna rešitev pa temelji na frameworku SpringBoot. Za nemoteno delovanje aplikacije je potreben vtičnik Maven. Za prikaz podatkov smo uporabljali tehnologijo JSP, ki za delovanje uporablja jezik Java.

Naš izbor integriranega razvojnega okolja je IntelliJ Idea, ki nudi največ funkcionalnosti in je za naše pojme najbolj pregleden. Celotna spletna stran je po statistiki GitHuba sestavljena iz: Java (83.2%), CSS (11.9%) in JavaScripta (4.9%). Torej smo v rešitev vključili večji del jezikov, ki smo jih spoznali tekom prvega letnika na smeri Informatika in tehnologije komuniciranja.



SLIKA 3: ODBTOTKI JEZIKOV

Arhitekturna zasnova aplikacije je razvidna iz E-R modela. Ključni entiteti sta torej dogodek in objekt. Primarno se spletna stran odpre na povezavi index.jsp, kjer so pregledno podane splošne informacije o problemu, naši podatki, nekaj povezav na zunanje strani, povezane z našo tematiko, izjemno pregledna »galerija« lokacij, ki po kliku na določeno sliko preusmeri na stran s filtriranimi dogodki, forme za prijavo, registracijo in kontakt.

Po kliku na željeno skupino dogodkov (restavracije, klubi, narava, kultura, šport), se izpiše pregleden seznam vseh dogodkov, ki ustrezajo filtru (naslovi povezav: restavracije.jsp ipd.). Po kliku na skupino dogodkov AKTUALNO, se izpišejo vsi dogodki vseh skupin. Vsaka skupina dogodkov vsebuje 3 poglede (sortirano po datumu, objektu ali strnjen pogled objektov na zemljevidu).

Po kliku na naziv željenega dogodka se odpre še podstran s podrobnostmi, ki v parameter prejme id izbranega dogodka (naslov povezave: podrobnosti.jsp/?id=x). Tako se s preglednim »gridom« izpišejo podrobnosti posameznega dogodka, kot so: daljši opis, naziv objekta, datum in ura začetka, datum in ura konca, kapaciteta in vstopnina, zanimanje za dogodek, galerija slik, lokacija na zemljevidu in komentarji).

V osnovi sta možna 2 tipa uporabnikov: prvi je neprijavljen uporabnik, ki ima osnovni pregled nad dogodki in stranjo, drugi pa uporabnik administrator, ki lahko dodaja dogodke, hkrati pa ureja le tiste dogodke, ki jih je dodal sam.

Do administratorja strani uporabnika vodi gumb »registracija« na dnu prve strani. Po kliku na gumb, je uporabnik preusmerjen na stran s formo (registracija.jsp), ki od uporabnika zahteva osnovne podatke. Tako se preko imena podjetja unikatno ustvari uporabniško ime, ki se nato posreduje uporabniku na mail. S tem mu je omogočen dostop do strani »dodajanje dogodka«, »moji dogodki« in »urejanje dogodka«.

Dodajanje dogodka je implementirano z osnovno formo, ki sprejme podatke o dogodku ter fotografije le-tega.

Stran z mojimi dogodki je praktično identična ostalim seznamom dogodkov, le da so na seznam vključeni samo dogodki, ki jih je dodal uporabnik sam. Po kliku na željenega se odpre stran za urejanje, ki je praktično identična strani »dodajanje dogodka«, le da ima v vnosnih poljih že vključene podatke izbranega dogodka.

S temi lastnostmi in funkcionalnostmi je uporaba strani izjemno poenostavljena in omogoča preprosto razumevanje klikov ter urejanj posameznih komponent.



SLIKA 4: LENT

GLAVNE KOMPONENTE, METODE

1. INDEX.JSP (GLAVNA STRAN) :

- Glavna stran, po kateri je omogočeno brskanje po strani s pomočjo hash-ov, prav tako pa vsebuje tudi povezave do ostalih spletnih strani. Z uporabo JSTL-ja se stran in povezave rahlo spremenijo, glede na avtoriteto (če je uporabnik prijavljen ali pa je anonimen).
- Vsebuje »portfolio«, na katerem so razdeljeni dogodki po kategorijah, torej filter, preko katerih se dostopa do seznama dogodkov, glede na tip objekta.
- Prijava je narejena s pomočjo SpringSecurity-a. S konfiguracijo EmployeeSecurityConfiguration, so določeni jsp-ji dostopni vsem, za nekatere pa je potrebna avtoriteta. Po registraciji, ki se nahaja na svoji strani, se pošlje mail tistemu, ki se je ravno registriral. Potem se lahko prijavi v sistem na koncu strani. Odjava je implementirana v menijski vrstici, ali pa na indexu, na mestu, kjer se je prej nahajala prijava.

2. REGISTRACIJA :

- Vsebuje enostavno formo, v katero se vnesejo podatki uporabnika in podatki o objektu.
- Če uporabnik uspešno izpolni vsa potrebna polja, potem se njegovi podatki shranijo v našo bazo podatkov, iz katere se kasneje bere pri prijavi.

3. STRANI, DOSTOPNE PREKO FILTROV NA GLAVNI STRANI :

- Na tej strani se odprejo sezname dogodkov, glede na izbran tip objekta, ali pa vsi, če je izbrana opcija »aktualno«. Te strani so drugače identične, in imajo iste funkcionalnosti. Za odpiranje seznama se uporabi metoda GetDogodekByFK(x), ta bere iz baze, in najde tip objekta po tujem ključu v tabeli.
- Odpre se seznam dogodkov, kjer pa je napisano ime dogodka, kraj, datum in slika dogodka, za lepši izgled seznama.
- Na tej strani sta tudi opciji za razvrščanje dogodkov, po datumu in po imenu objekta. To se pa implementira z identično metodo prejšnji, vendar se za to uporabi drugačen sql stavek (ORDER BY), ki razvrsti seznam po določenem parametru.

- Klik na dogodek v seznamu te poveže s stranjo »podrobnosti«.
- Opcija pogleda vseh objektov, ki gostijo dogodke na zemljevidu

4. PODROBNOSTI :

- Na strani podrobnosti, je napisan kratek opis dogodka, lokacija, datum, vstopnina in kapaciteta.
- Implementiran je »Like« gumb, kateri pokaže zanimanje za dogodek, ki je implementiran z jQuery-jem.
- Odpre se Galerija slik, ki so povezana na dogodek. Berejo se iz baze, odprejo pa se z JSTL stavkom »forEach«, ki prebere iz baze byte-array-e. Preko base64 encoder-ja pa se ta array spremeni v človeku vidno sliko.
- Glede na objekt, v katerem se dogodek izvaja, se na zemljevidu kreira marker, ki označuje lokacijo objekta
- Na tej strani so tudi implementirani komentarji, ki se shranijo v bazo, za vsak komentar pa tudi všeček, možnost odgovora in pa prijava zlorabe.

5. DODAJANJE DOGODKA :

- Po uspešni prijavi se odpre stran za dodajanje dogodka. Spet normalna forma, preko katere se dogodek shrani v bazo. Dogodek ima tuj ključ objekta in uporabnika, tako da, je vsak dogodek vezan na določen objekt oziroma uporabnika.
- Implementirano je tudi dodajanje slik na dogodek.

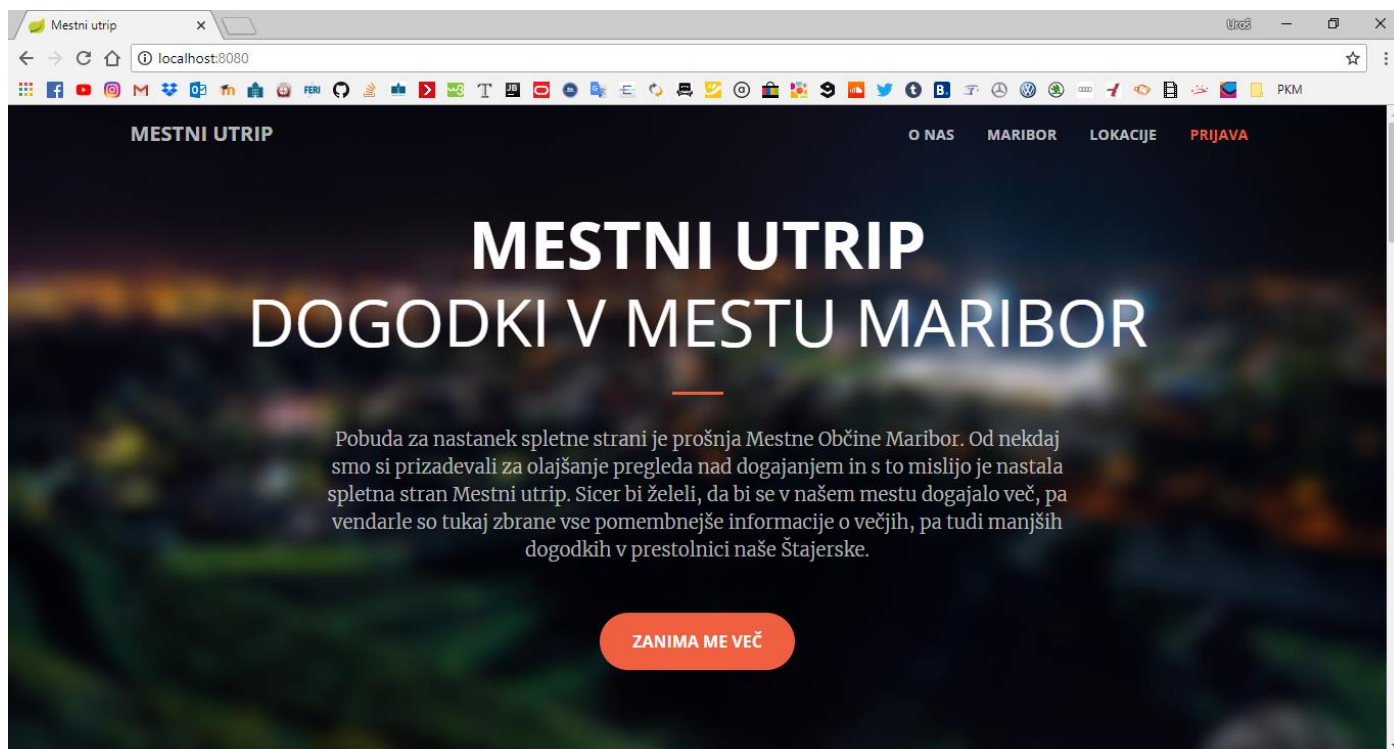
6. MOJI DOGODKI :

- Odpre se seznam dogodkov, ki jih je prijavljen uporabnik objavil. Uporabljena je zelo podobna metoda, kot na prejšnjih seznamih, vendar, se tu išče po tujem ključu uporabnika.

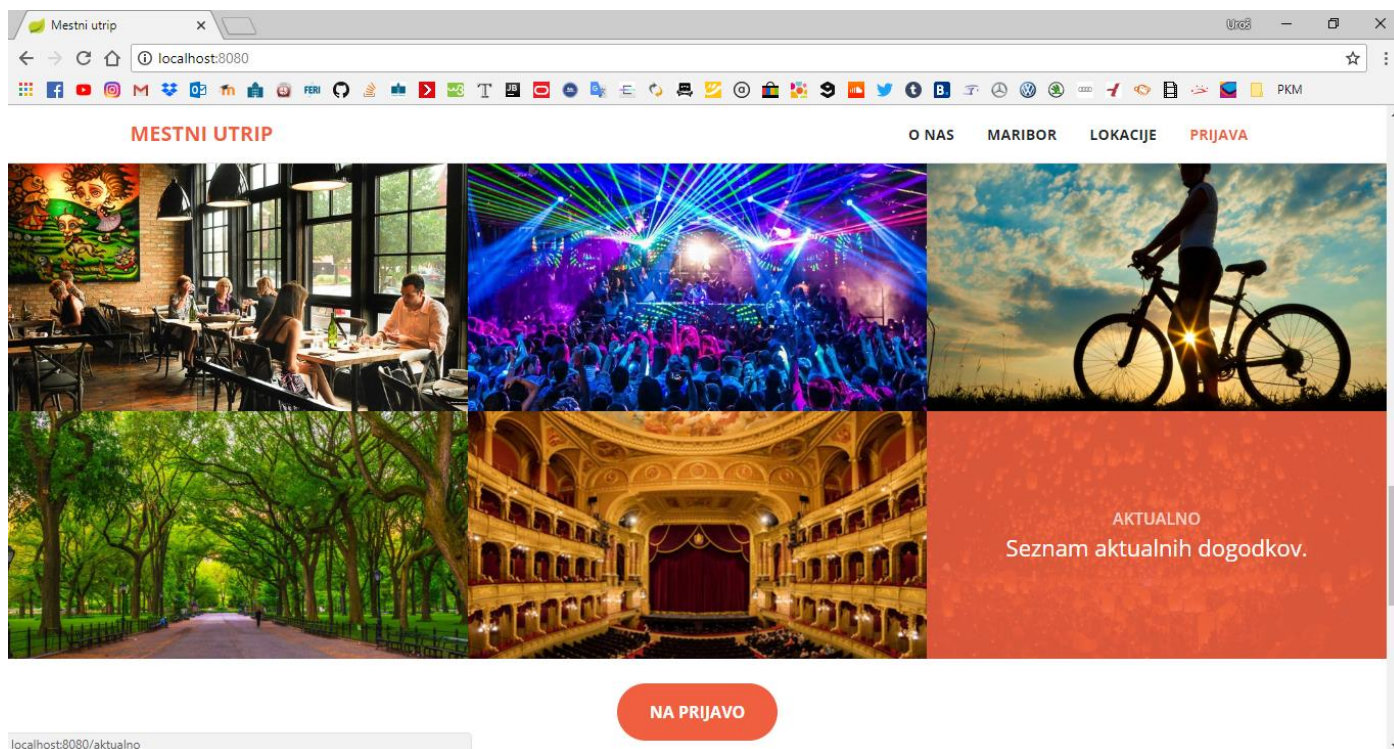
7. UREJANJE DOGODKOV :

- Ko se izbere dogodek, na strani mojiDogodki, se odpre forma, ki je identična formi za dodajanje, vendar se na tej le ureja, že obstoječi dogodek.
- Dogodek se lahko na tej strani tudi izbriše.

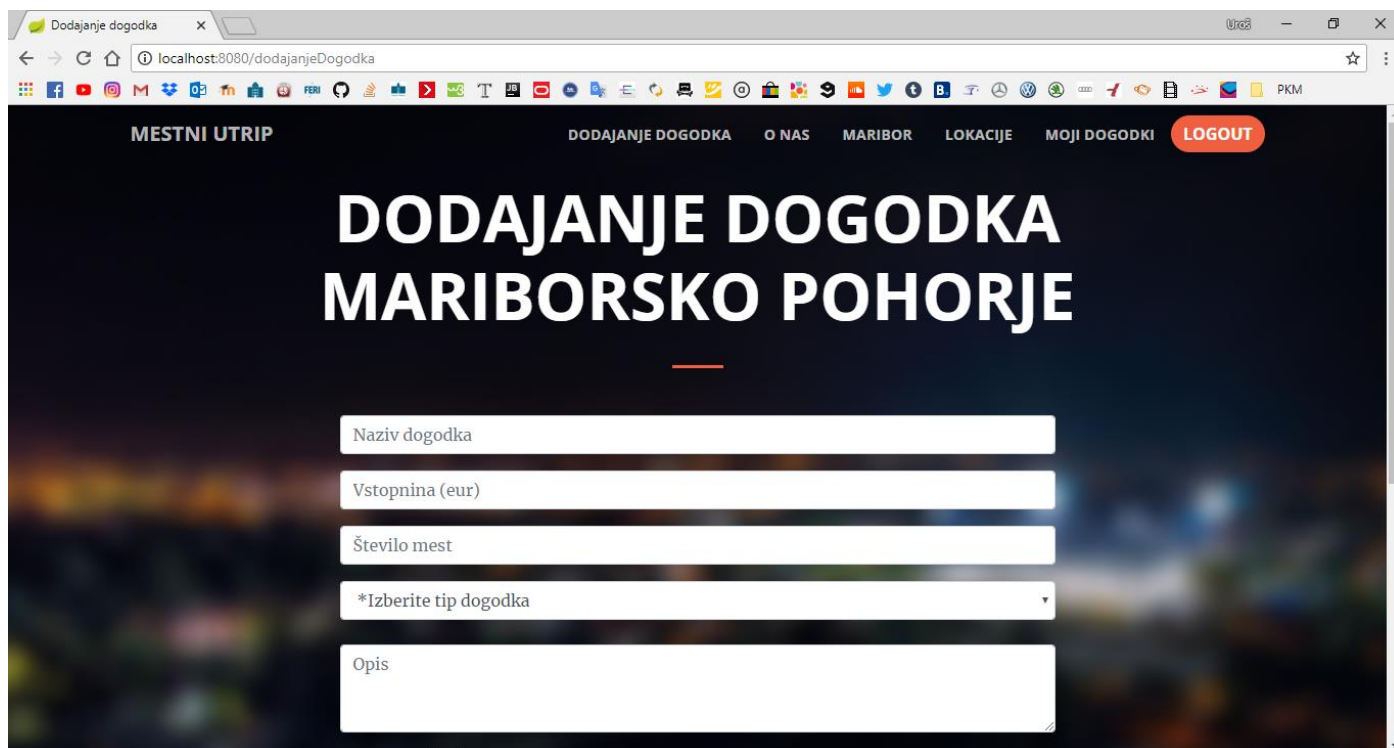
UPORABNIŠKI VMESNIK APLIKACIJE



SLIKA 5: POZDRAVNA STRAN INDEX.JSP



SLIKA 6: GALERIJA FILTRIRANIH SEZNAMOV



MESTNI UTRIP DODAJANJE DOGODKA O NAS MARIBOR LOKACIJE MOJI DOGODKI **LOGOUT**

DODAJANJE DOGODKA MARIBORSKO POHORJE

Naziv dogodka

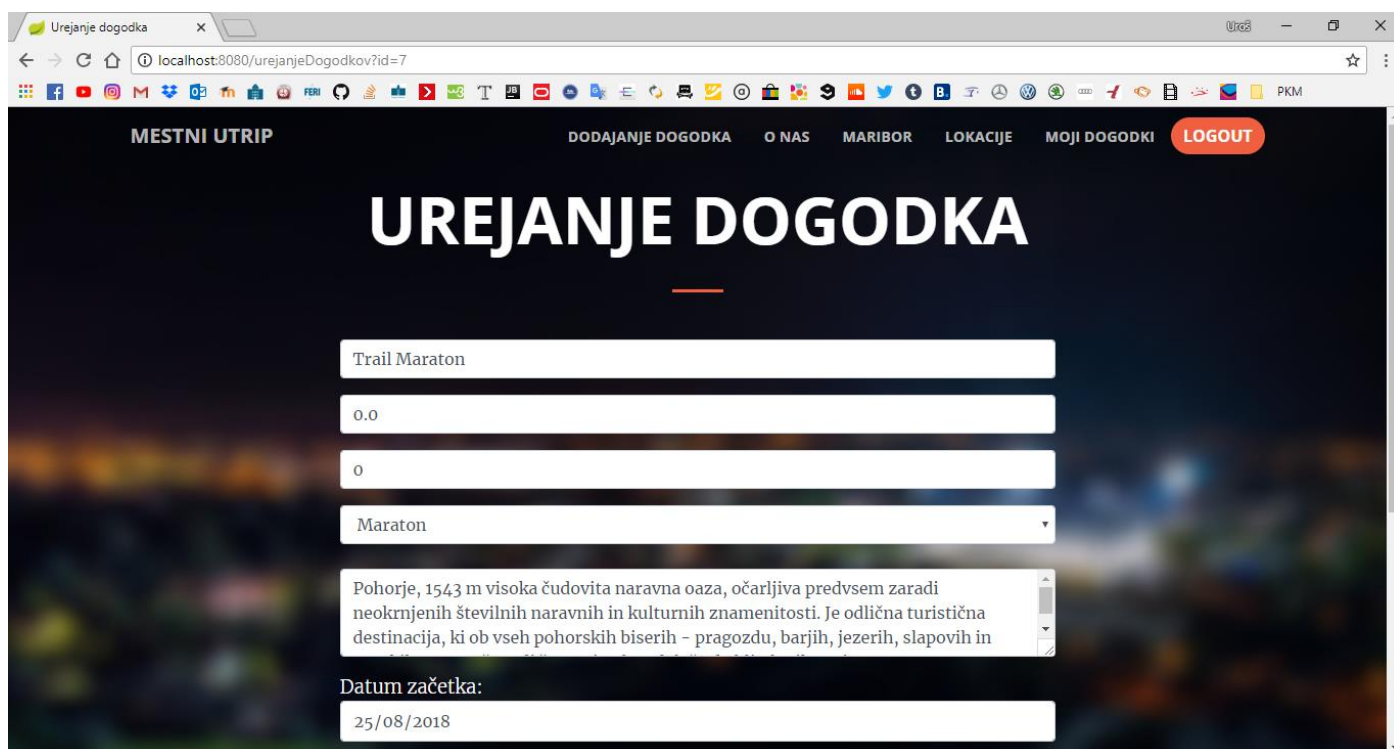
Vstopnina (eur)

Število mest

*Izberite tip dogodka

Opis

SLIKA 7: DODAJANJE DOGODKA



MESTNI UTRIP DODAJANJE DOGODKA O NAS MARIBOR LOKACIJE MOJI DOGODKI **LOGOUT**

UREJANJE DOGODKA

Trail Maraton

0.0

0

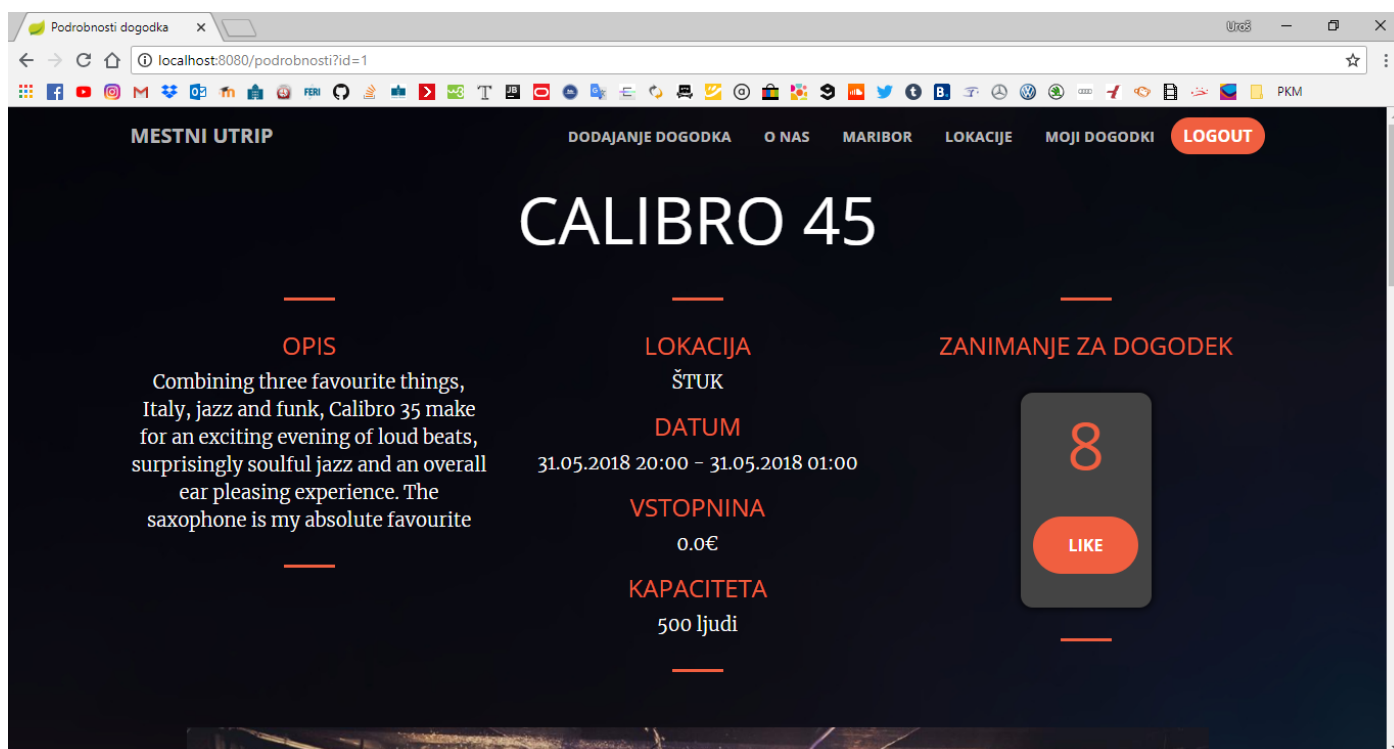
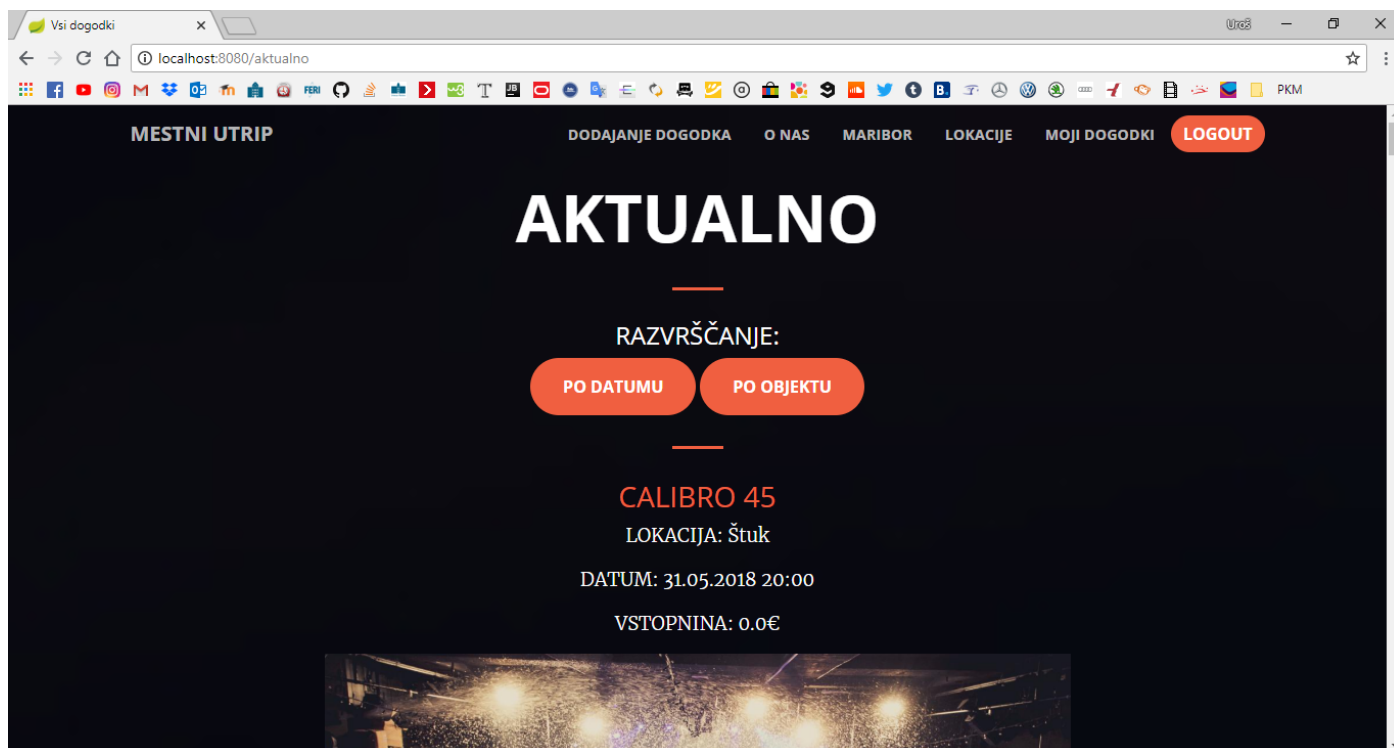
Maraton

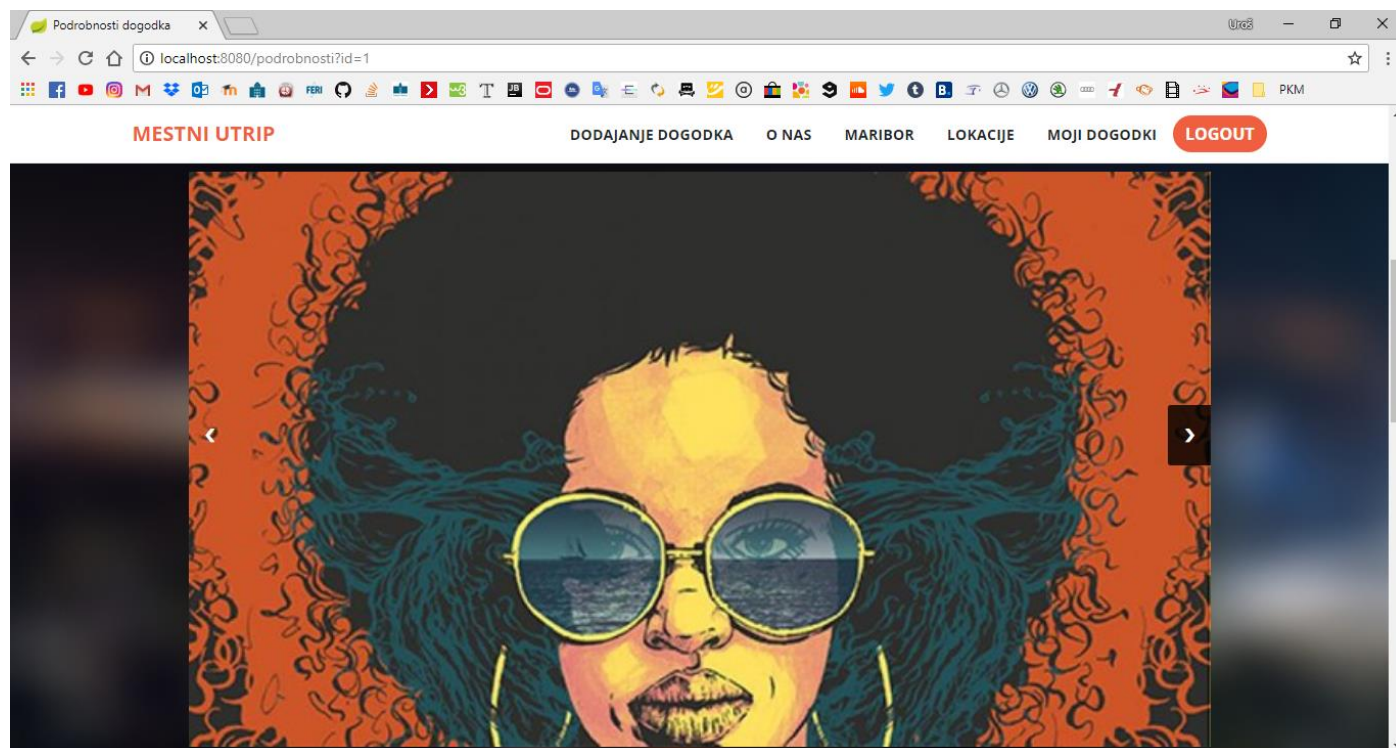
Pohorje, 1543 m visoka čudovita naravna oaza, očarljiva predvsem zaradi neokrnjenih številnih naravnih in kulturnih znamenitosti. Je odlična turistična destinacija, ki ob vseh pohorskih biserih - pragozdu, barjih, jezerih, slapovih in

Datum začetka:

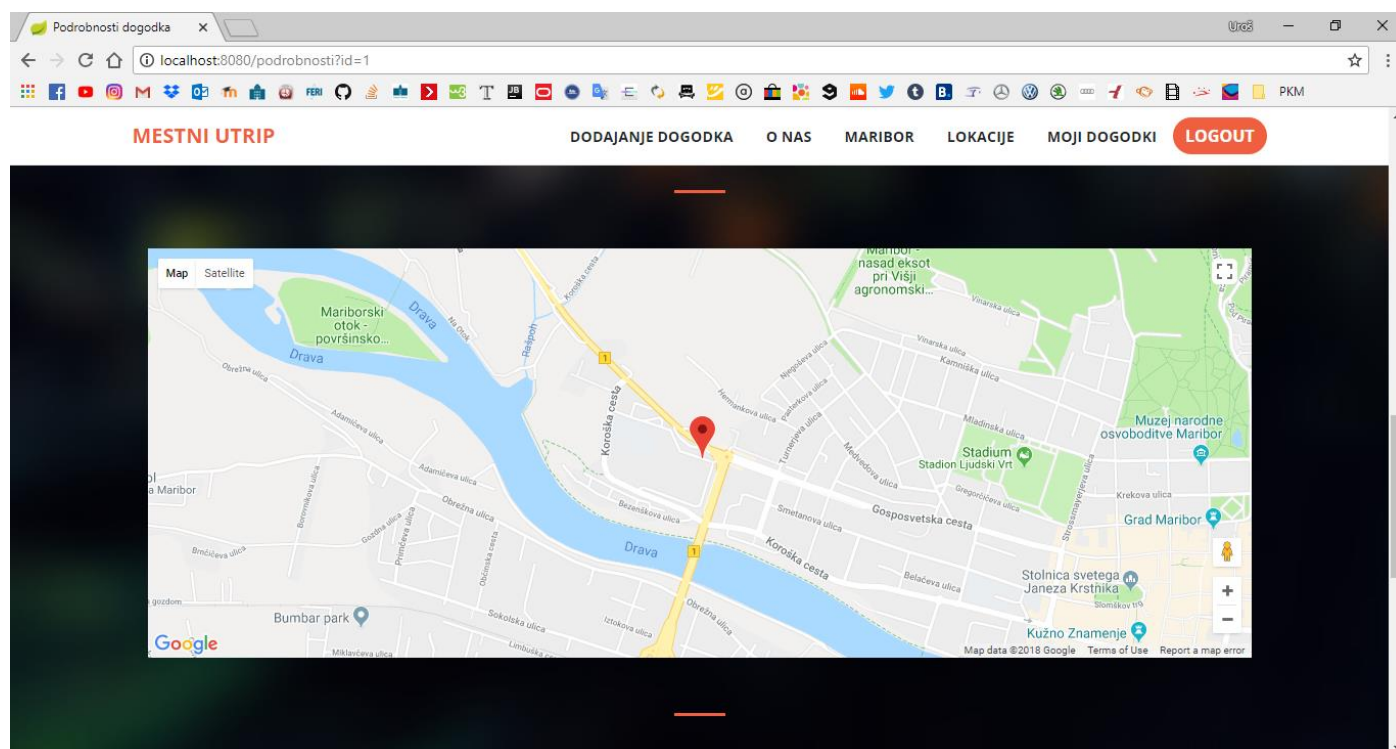
25/08/2018

SLIKA 8: UREJANJE ŽELJENEGA DOGODKA

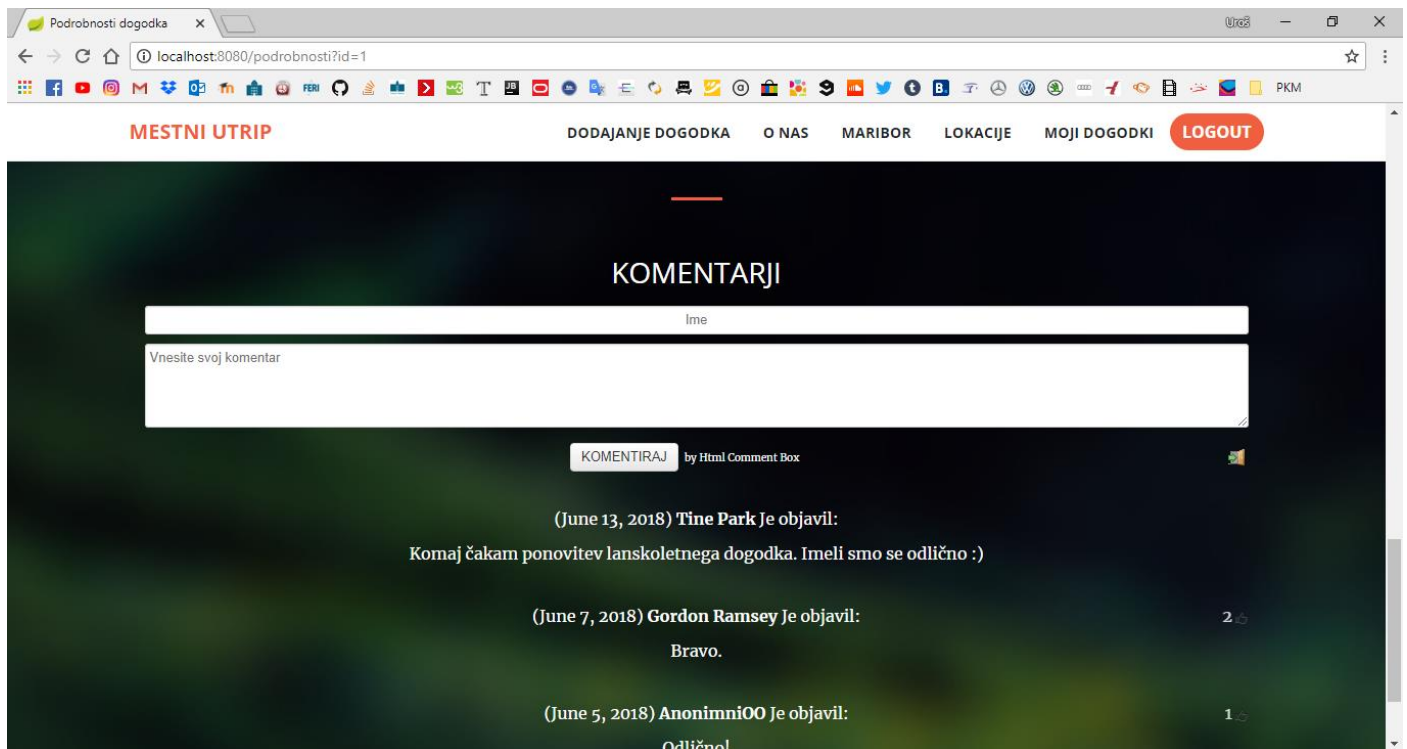




SLIKA 11: GALERIJA SLIK PRI PODROBNOSTIH

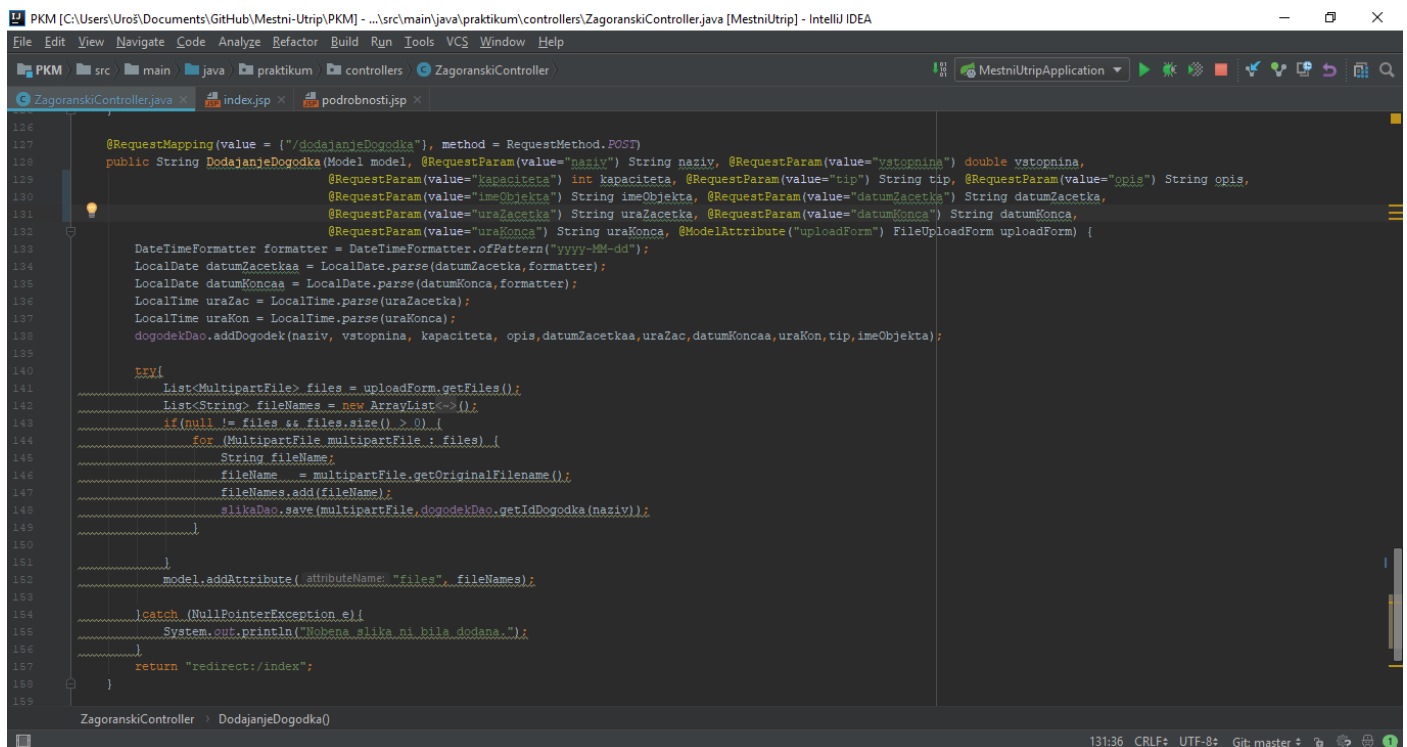


SLIKA 12: ZEMLJEVID PRI PODROBNOSTIH

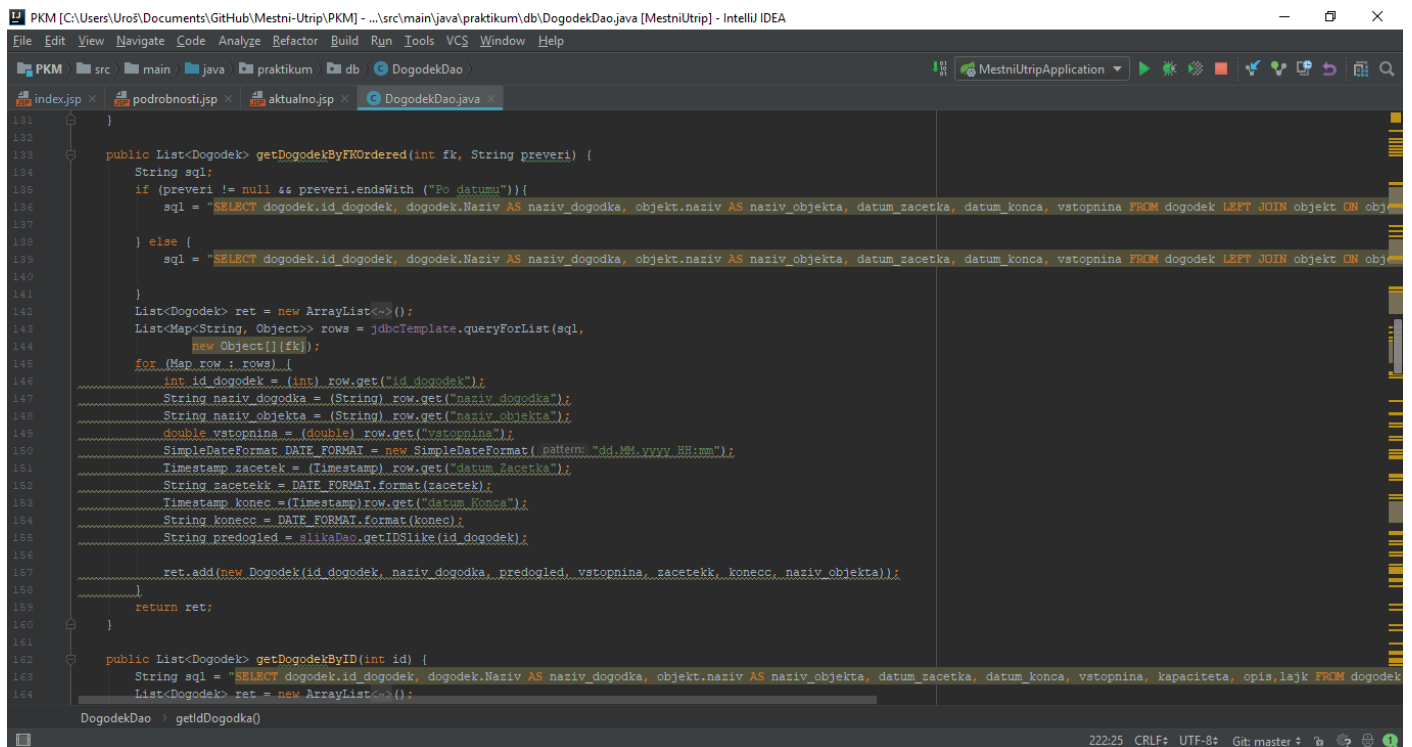


SLIKA 13: KOMENTARJI

POMEMBNEJŠI IZSEKI KODE



SLIKA 14: CONTROLLER DODAJANJE DOGODKA

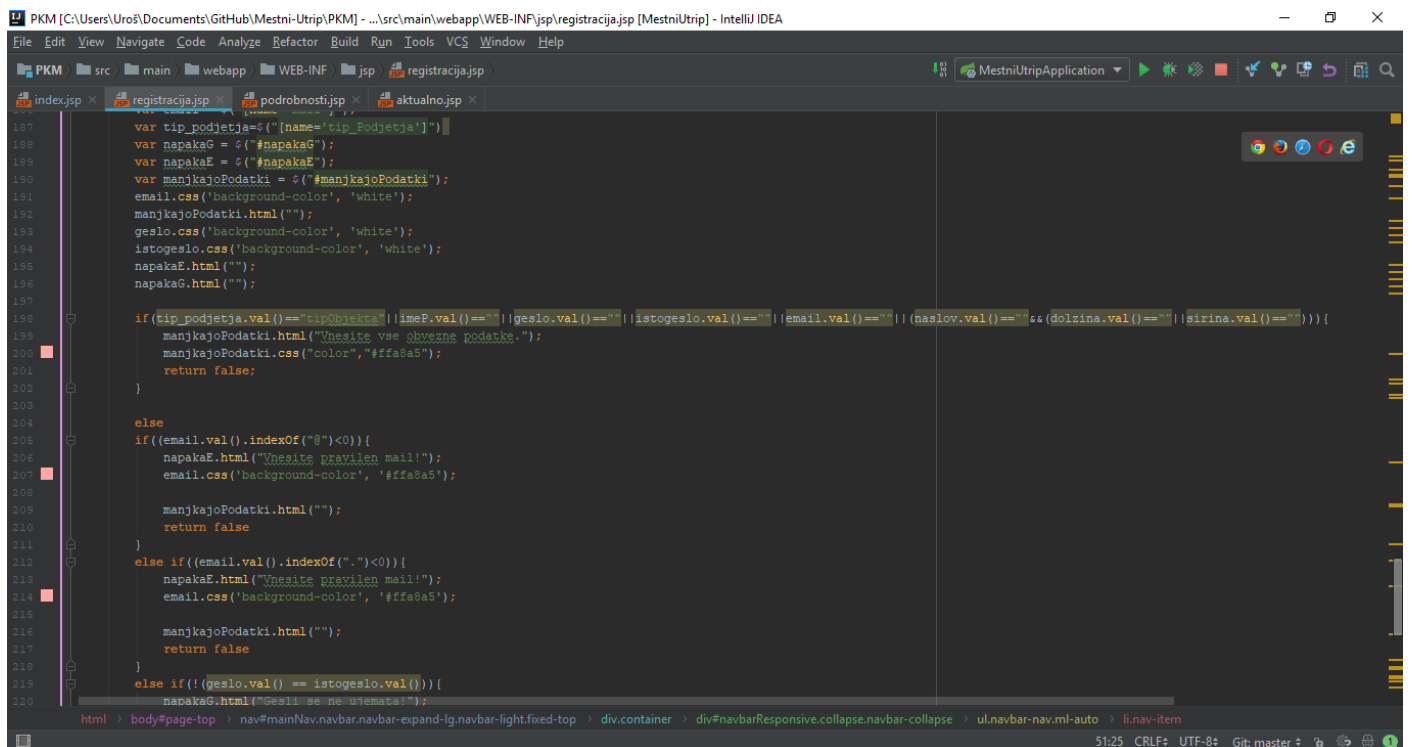


```

131 }
132
133 public List<Dogodek> getDogodekByFKOrdered(int fk, String preveri) {
134     String sql;
135     if (preveri != null && preveri.endsWith("Po datumu")) {
136         sql = "SELECT dogodek.id_dogodek, dogodek.Naziv AS naziv_dogodka, objekt.naziv AS naziv_objekta, datum_zacetka, datum_konca, vstopnina FROM dogodek LEFT JOIN objekt ON obje";
137     } else {
138         sql = "SELECT dogodek.id_dogodek, dogodek.Naziv AS naziv_dogodka, objekt.naziv AS naziv_objekta, datum_zacetka, datum_konca, vstopnina FROM dogodek LEFT JOIN objekt ON obje";
139     }
140     List<Dogodek> ret = new ArrayList<>();
141     List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql,
142         new Object[] {fk});
143     for (Map row : rows) {
144         int id_dogodek = (int) row.get("id_dogodek");
145         String naziv_dogodka = (String) row.get("naziv_dogodka");
146         String naziv_objekta = (String) row.get("naziv_objekta");
147         double vstopnina = (double) row.get("vstopnina");
148         SimpleDateFormat DATE_FORMAT = new SimpleDateFormat("dd.MM.yyyy.HH:mm");
149         Timestamp zacetek = (Timestamp) row.get("datum_zacetka");
150         String zacetek = DATE_FORMAT.format(zacetek);
151         Timestamp konec = (Timestamp) row.get("datum_konca");
152         String konec = DATE_FORMAT.format(konc);
153         String predogled = siikaDao.getIdSiika(id_dogodek);
154         ret.add(new Dogodek(id_dogodek, naziv_dogodka, predogled, vstopnina, zacetek, konec, naziv_objekta));
155     }
156     return ret;
157 }
158
159 public List<Dogodek> getDogodekByID(int id) {
160     String sql = "SELECT dogodek.id_dogodek, dogodek.Naziv AS naziv_dogodka, objekt.naziv AS naziv_objekta, datum_zacetka, datum_konca, vstopnina, kapaciteta, opis,lajk FROM dogodek";
161     List<Dogodek> ret = new ArrayList<>();
162     DogodekDao.getIdDogodka()

```

SLIKA 15: SORTIRANO BRANJE DOGODKA IZ BAZE

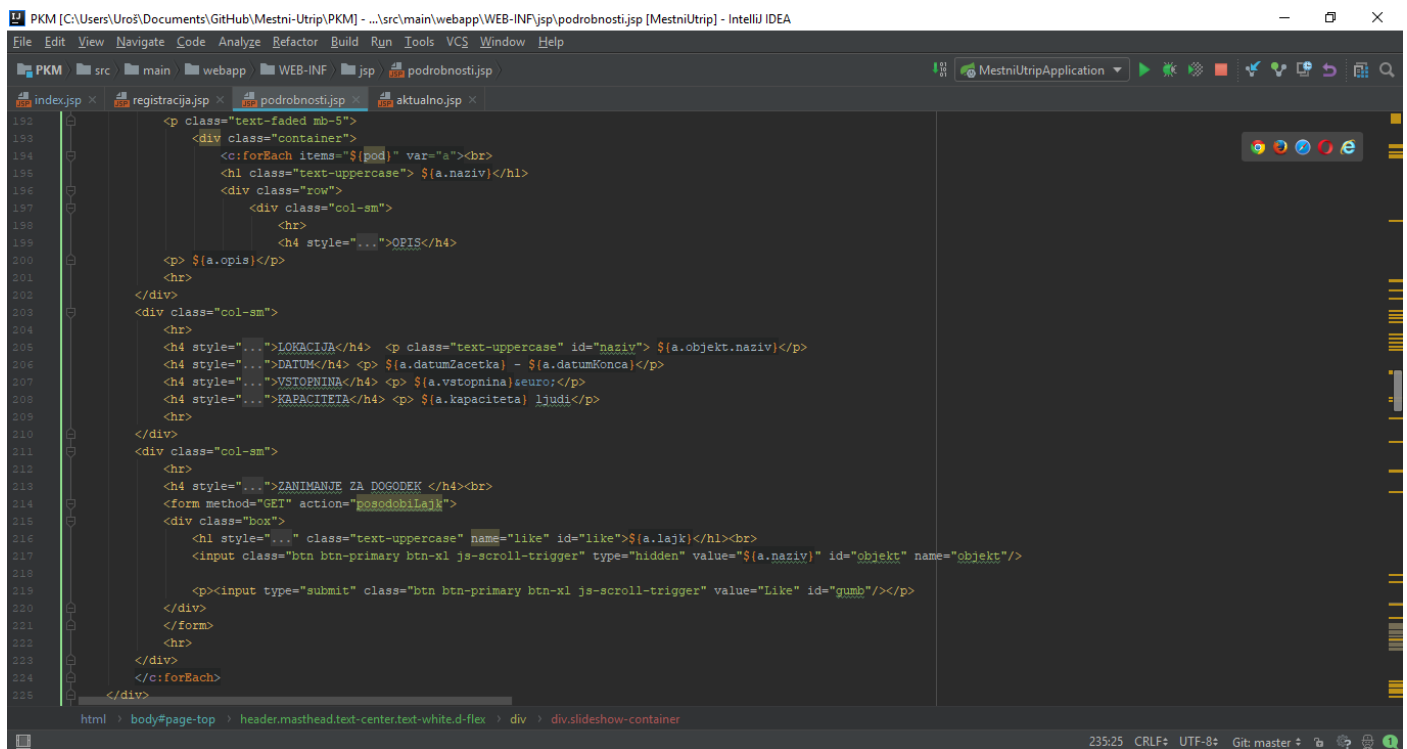


```

187 var tip_podjetja=$("#tip_Podjetja");
188 var napakaG=$("#napakaG");
189 var napakaE=$("#napakaE");
190 var manjkajoPodatki=$("#manjkajoPodatki");
191 email.css('background-color','white');
192 manjkajoPodatki.html("");
193 geslo.css('background-color','white');
194 istogeslo.css('background-color','white');
195 napakaE.html("");
196 napakaG.html("");
197
198 if(tip_podjetja.val()==="tipObjekta"||imeF.val()===""||geslo.val()===""||istogeslo.val()===""||email.val()===""||naslov.val()===""||dolzina.val()===""||sirina.val()==="){
199     manjkajoPodatki.html("Vnesite vse obvezne podatke.");
200     manjkajoPodatki.css("color","ffa5a5");
201     return false;
202 }
203
204 else
205 if((email.val().indexOf("@")<0)){
206     napakaE.html("Vnesite pravi email");
207     email.css('background-color','ffa5a5');
208     manjkajoPodatki.html("");
209     return false;
210 }
211 else if((email.val().indexOf(".")<0)){
212     napakaE.html("Vnesite pravi email");
213     email.css('background-color','ffa5a5');
214     manjkajoPodatki.html("");
215     return false;
216 }
217 else if(! (geslo.val() == istogeslo.val())){
218     napakaG.html("Gesli se ne ujemata!");
219 }

```

SLIKA 16: JQUERY PRI REGISTRACIJI



The screenshot shows the IntelliJ IDEA IDE with a JSP file named `podrobnosti.jsp` open. The code uses JSTL (Java Standard Tag Library) to iterate over a collection of objects and display their details. The code is as follows:

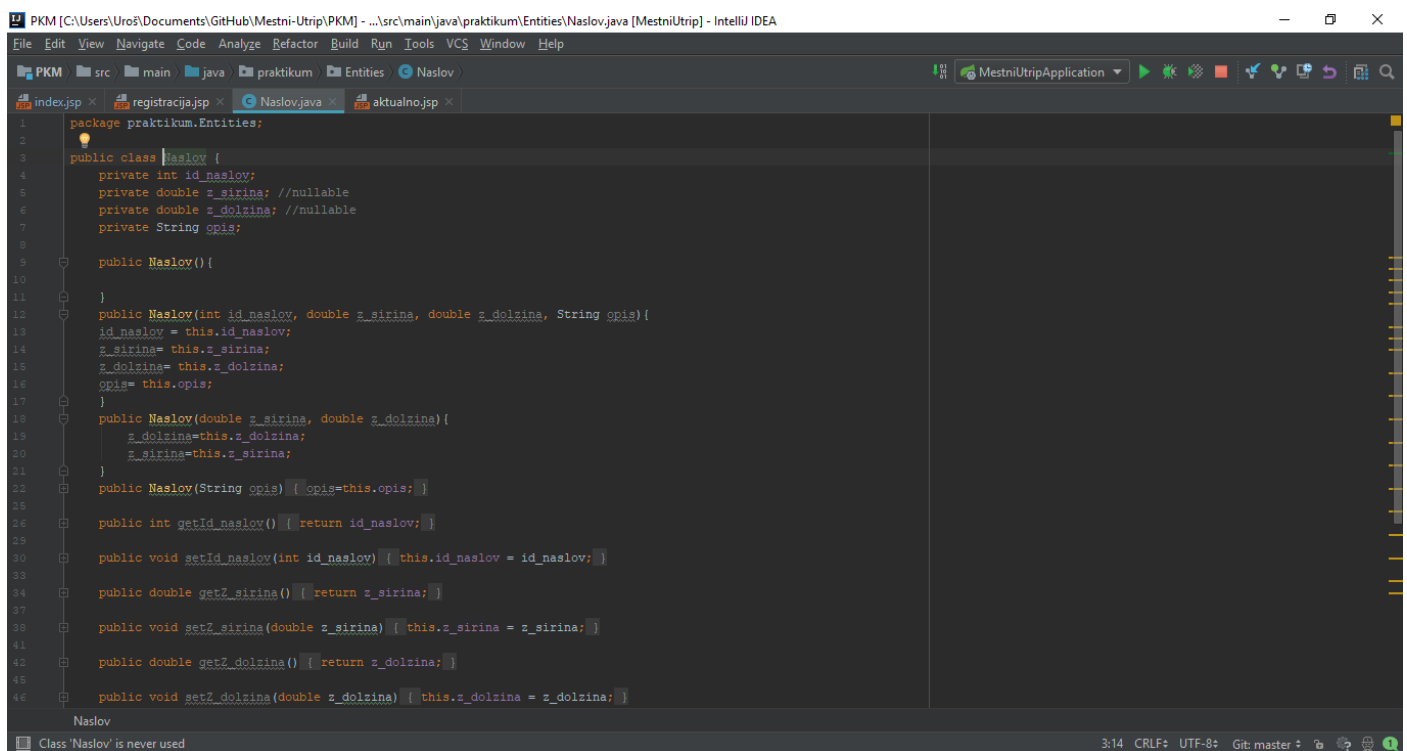
```

192 <p class="text-faded mb-5">
193   <div class="container">
194     <c:forEach items="${pod}" var="a"><br>
195       <h1 class="text-uppercase"> ${a.naziv}</h1>
196       <div class="row">
197         <div class="col-sm">
198           <hr>
199           <h4 style="...">OPIS</h4>
200           <p> ${a.opis}</p>
201         </div>
202       </div>
203       <div class="col-sm">
204         <hr>
205         <h4 style="...">LOKACIJA</h4> <p class="text-uppercase" id="naziv"> ${a.objekt.naziv}</p>
206         <h4 style="...">DATUM</h4> <p> ${a.datumZacetka} - ${a.datumKonca}</p>
207         <h4 style="...">VSTOPNINA</h4> <p> ${a.vstopnina}seuro</p>
208         <h4 style="...">KAPACITETA</h4> <p> ${a.kapaciteta} ljudi</p>
209       </div>
210     </div>
211     <div class="col-sm">
212       <hr>
213       <h4 style="...">ZANIMANJE ZA DOGODEK</h4><br>
214       <form method="GET" action="posodobilajk">
215         <div class="box">
216           <h1 style="..." class="text-uppercase" name="like" id="like"> ${a.lajk}</h1><br>
217           <input class="btn btn-primary btn-xl js-scroll-trigger" type="hidden" value="${a.naziv}" id="objekt" name="objekt"/>
218           <p><input type="submit" class="btn btn-primary btn-xl js-scroll-trigger" value="Like" id="gumb"/></p>
219         </div>
220       </form>
221     </div>
222   </div>
223 </c:forEach>
224 </div>

```

The status bar at the bottom shows the file path: `html > body#page-top > header.masthead.text-center.text-white.d-flex > div > div.slide-show-container`. The bottom right corner displays: `235:25 CRLF: UTF-8: Git: master`.

SLIKA 17: JSTL



The screenshot shows the IntelliJ IDEA IDE with a Java class named `Naslov` open. The code defines a class with private attributes and public methods. The code is as follows:

```

1 package praktikum.Entities;
2
3 public class Naslov {
4   private int id_naslov;
5   private double z_sirina; //nullable
6   private double z_dolzina; //nullable
7   private String opis;
8
9   public Naslov(){
10
11   }
12   public Naslov(int id_naslov, double z_sirina, double z_dolzina, String opis){
13     id_naslov = this.id_naslov;
14     z_sirina = this.z_sirina;
15     z_dolzina = this.z_dolzina;
16     opis = this.opis;
17   }
18   public Naslov(double z_sirina, double z_dolzina){
19     z_dolzina = this.z_dolzina;
20     z_sirina = this.z_sirina;
21   }
22   public Naslov(String opis) { opis = this.opis; }
23
24   public int getId_naslov() { return id_naslov; }
25
26   public void setId_naslov(int id_naslov) { this.id_naslov = id_naslov; }
27
28   public double getZ_sirina() { return z_sirina; }
29
30   public void setZ_sirina(double z_sirina) { this.z_sirina = z_sirina; }
31
32   public double getZ_dolzina() { return z_dolzina; }
33
34   public void setZ_dolzina(double z_dolzina) { this.z_dolzina = z_dolzina; }
35
36 }

```

The status bar at the bottom shows the file path: `Naslov`. The bottom right corner displays: `3:14 CRLF: UTF-8: Git: master`. A message at the bottom left states: `Class 'Naslov' is never used`.

SLIKA 18: JAVANSKI RAZRED

POROČILO TESTIRANJA

Testiranje smo izvedli tekom izdelave aplikacije, zato posebno testiranje na koncu ni bilo potrebno. Preizkusili smo vse mogoče funkcionalnosti in klike na naši strani, zato smo sto odstotno prepričani, da spletna stran deluje tako kot mora.



SLIKA 19: TESTIRANJE