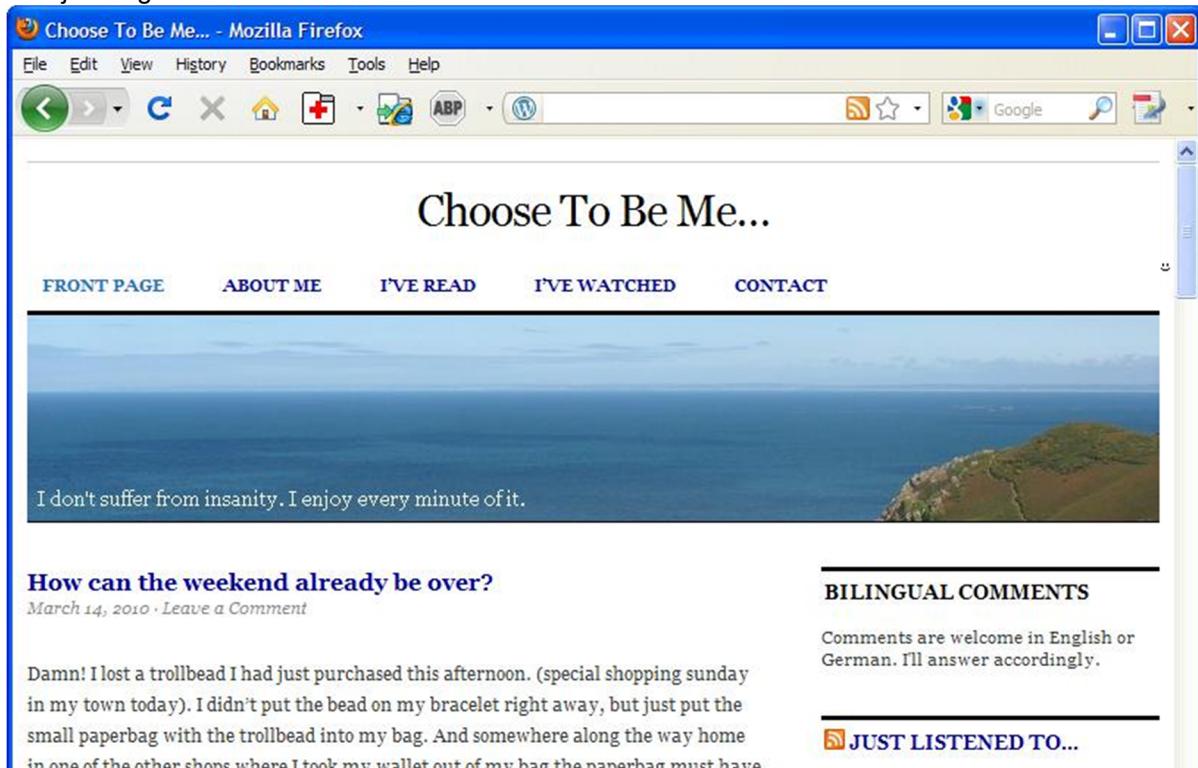


**Izrada aplikacije za blog platformu
sa pripadnom bazom podataka u MongoDB**
Izradili Lucija Valentić, Fran Vojković, Martina Gaćina,
Dario Bogović, Iva Tutiš

1.Uvod

Blog platforma je web stranica koja – halik dnevniku – omogućuje njenom korisniku pisanje tekstova o proizvoljnim temama, koji se na stranici prikazuju u obrnutom kronološkom redu (tako da se najnoviji objavljeni tekstovi pojavljuju prvi po redu). Doduše, za razliku od dnevnika, blog platforma je javna – dakle svaki tekst koji registrirani korisnik objavi je dostupan javnosti za pregled, te je dostupan drugim korisnicima blog – platforme za komentiranje.

Primjer Blog Platforme:



Naša aplikacija će biti oblika *MAB* („*Multi-Author Blog*“) jer će tekstovi („*posts*“) na njoj biti djelo ne jednog, nego grupi više registriranih korisnika blog platforme, u koju se novi potencijalni korisnik može lako uključiti korištenjem *log-in* forme.

Pritom će svaki korisnik blog-platforme imati svoj „dnevnik“, to jest svaki registrirani korisnik koji je objavljivao tekstove će imati vlastitu web-stranicu na kojoj će se prikazivati svi njegovi tekstovi, zajedno s komentarima na njih.

Informacije o svakom tekstu te popratne informacije koje idu uz tekst (vrijeme objave, naslov, komentari, podaci o autoru...), te informacije o registriranim korisnicima (username, šifre...) će biti pohranjeni kao baza podataka u programu za baze podataka MongoDB.

Sama aplikacija je napravljena u programskom jeziku Java (koji se spaja na MongoDB te preko upita obrađuje dobivene podatke), budući Java kao objektno-orientirani jezik nam lako pruža mogućnosti učitavanja podataka.

Front-end dio aplikacije, to jest korisničko sučelje, je izvedeno sa kombinacijom HTML+CSS programskih jezika.

2. Podatkovni model

Odlučili smo kreirati dvije kolekcije koje će tvoriti našu bazu podataka:

- Kolekcija *login*, koja će spremati podatke o svim registriranim korisnicima na blog platformi. (autor je određen sa svojim korisničkim imenom)

```
{  
    "korisnicko_ime" : ,  
    "sifra" :  
}
```

- Kolekcija *tekstovi*, koja će spremati podatke o objavljenim tekstovima

```
//oblik podatka  
{  
    "text_id" : ,  
    "text" : ,  
    "naslov" : ,  
    "autor" : {  
        "korisnicko_ime" : ,  
        "sifra" :  
    },  
    "teme" : [  
    ],  
    "komentari" : [  
        {  
            "korisnicko_ime" : ,  
            "text" :  
        },  
        {  
            "korisnicko_ime" : ,  
            "text" :  
        }  
    ],  
    "slike" : [  
    ]  
}  
  
//kreiranje kolekcije  
db.createCollection( "tekstovi" )
```

3. Punjenje baze podataka

Bazu podataka smo napunili sa podacima koristeći klasičnu naredbu db.collection_name.insert({...}).

Primjer naredbe za punjenje kolekcije login:

```
db.login.insert{  
    "korisnicko_ime" : "Luka",  
    "sifra" : "lukinasifra"  
};
```

Primjer naredbe za punjenje kolekcije tekstovi:

```
db.tekstovi.insert({  
    "text_id" : 4,  
    "text" : "Sve je počelo tamo negdje još u vrtiću i protegnulo se kroz osnovnu školu i  
    nastupe na Lidranu. Zatim su se u Miocu moji javni nastupi prorijedili, da bih im se  
    ponovno vratila tijekom faksa i eSTUDENTa. Moja priča s javnim nastupima je  
    eskalirala tijekom proteklih nekoliko godina kada sam se zaposlila kao trenerica  
    poslovnih vještina zbog čega "pričam pred ljudima" barem jedan dan u tjednu.",  
    "naslov" : "15 trikova kako steći samopouzdanje za javni nastup",  
    "autor" : {"id" : 1,"ime" : "Tea","sifra" : "teinasifra"},  
    "teme" : ["osobni razvoj","organizacija","trema","javni nastup","posao"],  
    "komentari" : [  
        {"korisnicko_ime" : "Ana","text" : "Ajme ja se bojim nastupati! >< "},  
        {"korisnicko_ime" : "Marko", "text" : "Ovo mi je spasilo život. Hvala."}  
    ],  
    "slike" : []  
});
```

Za konstrukciju naših podatka s kojima će se baza puniti smo koristili nasumične objave sa blog-a <https://markozupanic.hr/blogovi/> (i malo mašte.)

Iz tih podataka, s naredbama za kreiranje i stavljanje podataka u kolekcije smo konstruirali datoteke *login.csv* i *test.csv*, sa kojima mongo može automatski napuniti pripadne kolekcije.

3.2 Pokretanje baze podataka

The screenshot shows two panes of the JSON Editor Online interface. The left pane displays a MongoDB document with the following code:

```
1 db.createCollection( "tekstovi" )
2
3 db.tekstovi.insert({
4   "autor" : "Martina",
5   "naslov" : "Napredne baze podataka",
6   "temo" : [
7     "fakultet",
8     "baze podataka",
9     "PMF",
10    "PMF-MO"
11  ],
12  "text" : "Kolegij napredne baze podataka se održava na Prirodoslovno-matematičkom fakultetu, Matematičkom odjeljku u Zagrebu. Održava se na diplomskim studijima Računarstvo i matematika te Informatika i matematika - nastavnički smjer, u ljetnom semestru.",
13  "komentari" : [
14    {
15      "autor" : "Iva",
16      "text" : "Napredne baze podataka su smeđe!"
17    },
18    {
19      "autor" : "Fran",
20      "text" : "Nisam imao interneta za prvu zadaću."
21    }
22  ],
23  "slike" : [
24
25  ]
26},
27
28]
```

The right pane shows the JSON representation of the document:

```
object {0}
  (empty object)
```

Below the editor, a message bar states: "This site uses cookies to personalize ads. Information about your use of this site is shared with ad providers. By using this site, you agree to its use of cookies. [learn more](#)". The status bar at the bottom shows: "autor.json 0016 22/05/2020".

Pod prepostavkom da Docker i MongoDB nisu instalirani, pratiti poveznicu u Izvori(7.).

1. pokrenuti mongo sa `docker start npb-mongo`
2. kopirati dokumente test.csv i login.csv (u npr. na linuxima: /home/user/)
3. kopirati navedene datoteke u kontejner naredbama:
`docker cp test.csv npb-mongo:/home/`
`docker cp login.csv npb-mongo:/home/`
4. Pokrenuti bash shell naredbom: `docker exec -it npb-mongo bash`
5. učitati podatke u novu bazu, npb, kolekcije tekstovi i login:
`root@43180e0447ea:/# mongoimport -d npb -c tekstovi --file/home/test.csv`
`root@43180e0447ea:/# mongoimport -d npb -c login --file/home/login.csv`

4. Potrebni upiti u MongoDB

Za razvoj aplikacije smo determinirali potrebu za određenim upitim.

4.1 Nad kolekcijom *login*

- Dohvaćanje svih korisnika aplikacije
`db.login.find()`
- Dohvaćanje svih (različitih) imena korisnika aplikacije
`db.login.distinct("korisnicko_ime")`
- Dohvatiti šifru korisnika ako imamo njegovo ime
`db.login.findOne({ "korisnicko_ime" : "korisnicko_ime" }, { "_id":0, ".sifra" : 1 })`

)

4.2 Nad kolekcijom *tekstovi*

- Promijeni naslov teksta

```
db.tekstovi.update(  
  { "text_id" : text_id }, { $set: { "naslov": "naslov" } }  
)
```
- Promijeni temu teksta

```
db.tekstovi.update(  
  { "text_id" : text_id }, { $set: { "teme": [ "tema1", "tema2" , ...] } }  
)
```
- Dodaj novu temu

```
db.tekstovi.update(  
  { "text_id" : text_id }, { $push: { "teme": "tema" } }  
)
```
- Promijeni tekst
 - Ako dodajemo skroz novi tekst

```
db.tekstovi.update(  
  { "text_id" : text_id }, { $set: { "text": "Novi text" } }  
)
```
 - Ako zelimo samo nesto promijeniti u tekstu trebamo prvo dohvatiti tekst:

```
db.tekstovi.find(  
  { "text_id" : text_id }, { "_id":0, "text" : 1 }  
)
```
 - Zatim spremiti promijenjeni tekst:

```
db.tekstovi.update(  
  { "text_id" : text_id }, { $set: { "text": "Promijenjeni text" } }  
)
```
- Dodati novi komentar na određeni tekst:

```
db.tekstovi.update(  
  { "text_id" : text_id },  
  { $push: {  
    "komentari": { "korisnicko_ime" : "korisnicko_ime", text" : "Novi komentar" }  
  } }  
)
```
- Dohvatiti određeni tekst

```
db.tekstovi.find(  
  { "text_id" : text_id }  
)
```
- Dohvatiti sve komentare na određeni tekst

```
db.tekstovi.find(  
  { "text_id" : text_id }, { "_id":0, "komentari" : 1 }  
)
```
- Svi tekstovi određenog autora

```

db.tekstovi.find(
  { "autor.korisnicko_ime" : "korisnicko_ime" }
)

```

- Analizirati tekstove po temama (*Map/Reduce framework*)

U našoj bazi podatka će tekstovi biti raspodijeljeni po ukupno 6 tema:
biznis & karijera, marketing & SEO, digitalno poslovanje, osobni razvoj, trening & prehrana i putovanja.

Ukoliko želimo dohvatiti sve tekstove (identificirane sa *text_id*) koji imaju neku zadanu temu, dovoljno nam je koristiti Map/Reduce framework, koji će spremati listu identifikacija tekstova po temama u novu kolekciju *tekstovi_po_temama*

```

var mapFunction1 = function() {
  for(var i=0; i<this.teme.length; i++){
    var key = (this.teme[i]).product;
    var value = new Array(this.text_id);
    emit(key, value);
  };
};

var reduceFunction1 = function(key, values_list) {
  //values_list je lista jednocalnih listi.
  //zelimo umjesto toga imati listu text_id-eva
  var reduced_value=new Array();

  for (var i=0; i<values_list.length; i++){
    var current_value=values_list[i];
    reduced_value.push(current_value[0]);
  };
  return reduced_value;
};

db.tekstovi.mapReduce(
  mapFunction1,
  reduceFunction1,
  { out: {merge: "tekstovi_po_temama"}}
);

```

Sada imamo novu kolekciju s elementima oblika:

```

//stvara novu kolekciju "tekstovi_po_temama" kojoj je jedan član oblika
{
  "_id": tema,
  "value": [text_id_1, text_id_2,...]
}

```

Iz koje lako dohvativimo listu tekstova o nekoj temi sa:

```
db.tekstovi_po_temama.find({_id :"naša_tema"}, { _id:0, "value":1})
```

5. Pokretanje (*setup*) Aplikacije



Da bi uspješno pokrenuli aplikaciju u Javi, potrebni su nam neki popratni instalirani programi:

- *jdk*:
<https://www.oracle.com/java/technologies/javase-downloads.html>
- *jre*:
<https://www.java.com/en/download/>
- *apache maven*:
<https://maven.apache.org/download.cgi>
- *apache tomcat*:
<https://tomcat.apache.org/download-80.cgi>

Zatim, unutar direktorija u kojem se nalazi kod aplikacije, treba pokrenuti naredbu:

`mvn package`

Nakon toga će se tamo pojaviti datoteka koju trebamo premjestiti:

1. za pokretanje aplikacije: u folderu /apachetomcate, unutar poddirektorija /webapps kopirati stvorenu datoteku pa pokrenuti apache tomcat naredbom iz izvršnog direktorija:
`bin/startup.bat`
2. aplikacija se pokreće uz pomoć internet preglednika, preko linka:
`http://localhost:8080/blog/index.jsp`

Nakon što je aplikacija završi sa radom, apache tomcat se može isključiti uz pomoć naredbe:
`bin/shutdown.bat`

6. Izgled Aplikacije

6.1 Login form

6.1.1 Log-in postojećeg korisnika

The screenshot shows a 'Login' page with a light gray header containing the word 'Login'. Below the header is a vertical dark gray sidebar. The main content area has a white background. It contains the word 'Login' in bold, followed by two input fields: 'Username:' and 'Password:', each with a corresponding text input box. Below these fields is a dark gray 'Login' button. At the bottom of the page is a dark gray footer box containing the text 'Registriraj novog korisnika!'. Below the footer, there is a horizontal line of text: 'Lista registriranih korisnika i linkovi koji vode na popis blogova istih:'.

Below this, there is another identical login form structure, but it lacks the 'Username:' field. It also features a 'Password:' field, a 'Login' button, and a 'Registriraj novog korisnika!' footer.

Following this second form, there is a list of registered users:

- 1. Tea
- 2. Ana
- 3. Marko
- 4. Lucija
- 5. Jakov

Ukoliko se unese kriva šifra za već postojećeg korisnika, ili nepostojeće ime korisnika, aplikacija javlja grešku.

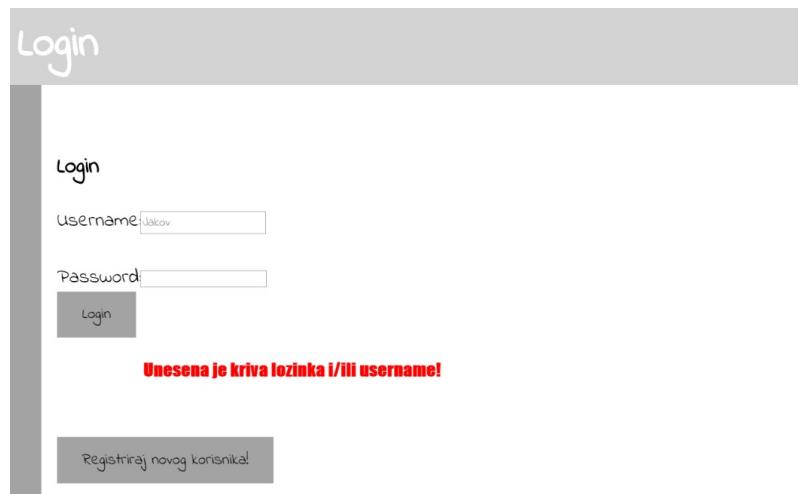
Login

Login

Username: Jakov

Password:

Unesena je kriva lozinka i/ili username!

A screenshot of a login interface. At the top, it says "Login". Below that is a vertical grey sidebar. On the right side, there are two input fields: "Username" containing "Jakov" and "Password" which is empty. Below the fields is a button labeled "Login". A red error message "Unesena je kriva lozinka i/ili username!" is displayed. At the bottom is a button labeled "Registriraj novog korisnika!".

6.1.2 Registracija novog korisnika

Regisriraj se

Username: Dario

Lozinka:

A screenshot of a registration interface. At the top, it says "Regisriraj se". Below that is a vertical grey sidebar. On the right side, there are two input fields: "Username" containing "Dario" and "Lozinka" containing "....". Below the fields is a button labeled "Registriraj me!". At the bottom is a button labeled "vrazi se na pocetnu!".

Korisnik se registrira upisom željenog usernamea i novog passworda.

Ukoliko već postoji korisnik sa zadanim username-om, aplikacija će javiti grešku.

U suprotnom, uspjeh.

Poruka

Uspješna registracija!

A screenshot of a success message interface. At the top, it says "Poruka". Below that is a vertical grey sidebar. On the right side, the text "Uspješna registracija!" is displayed. At the bottom is a button labeled "vrazi se na pocetnu!".

6.2 Pisanje tekstualne objave na blog-u korisnika (novi „post“)

Ukoliko smo mi registrirani korisnik, na svojoj stranici s listom blogova možemo dodati novi blog.

The image consists of three vertically stacked screenshots of a web-based blog application:

- Screenshot 1: Blogovi**

A header bar with the title "Blogovi". Below it, a list item "1. proba" is shown. At the bottom are three buttons: "vrati se na početnu!", "Dodaj novi blog", and "Log out".
- Screenshot 2: Novi Blog Post**

A header bar with the title "Novi Blog Post". The "Naslov" field contains the placeholder "Ovo je moj novi blog". The "Sadržaj" field contains the text "Moj novi blog!". Below these fields is a "Objavi!" button.
- Screenshot 3: Poruka**

A header bar with the title "Poruka". The main content area displays the message "Blog uspješno dodan!". At the bottom is a "vrati se na početnu!" button.

Po objavi, možemo vidjeti blog i naći ga ubuduće kroz listu blogova korisnika.

6.3 Dodavanje komentara

Na svaki blog (kao registrirani korisnik!) možemo ostaviti komentar.

Komentar

Sadržaj

Ovo je moj komentar!

Dodaj komentar!

Poruka

Komentar uspješno dodan

Vrati se na početnu!

Nakon uspješnog dodavanja komentara, isti se vidi ispod teksta blog-a.

6.4 Pogled na tekst (izgled *blog-a*)

Popis blogova (po korisniku) se može vidjeti u glavnom izborniku.

A screenshot of a web page showing a registration form. At the top right is a button labeled "Registriraj novog korisnika!". Below it is a list titled "Lista registriranih korisnika i linkovi koji vode na popis blogova istih:" followed by a numbered list of users:

1. Tea
2. Ana
3. Marko
4. Lucija
5. Jakov

Kliknemo li na nekog autora, te potom i na željenu temu bloga, možemo čitati tekst bloga, te vidjeti komentare pripadne tom blog-post-u.

A screenshot of a blog post page. The header says "Blog". The main content area shows a blog post titled "Moj novi blog" with an "uredi" button. Below the post is a section titled "Komentari:" containing one comment from "Dario" with the text "Ovo je moji komentar". At the bottom are two buttons: "Vrati se na početnu!" and "Dodaj novi komentar!".

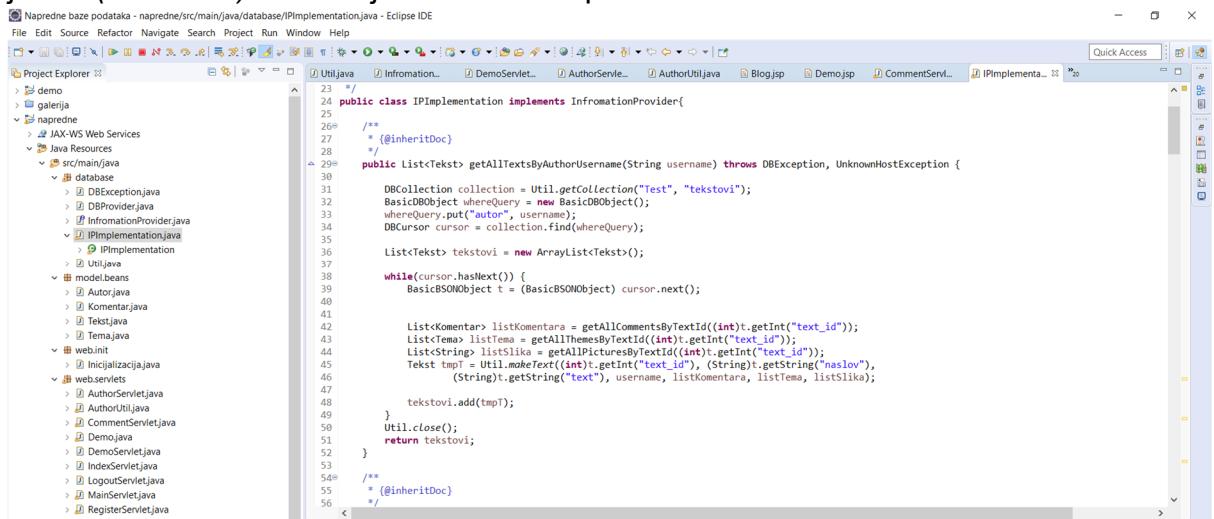
7.Kod u Javi (kod aplikacije)

Nakon što smo vidjeli izgled i „layout“ aplikacije, biti će nam jednostavnije razumjeti kod koji slijedi.

Kod je organiziran u 3 funkcionalna dijela. Jedan dio se bavi bazama podataka, jedan dio priprema podatke koji se trebaju prikazati u aplikaciji ili koje treba nekako spremiti u bazu podataka.

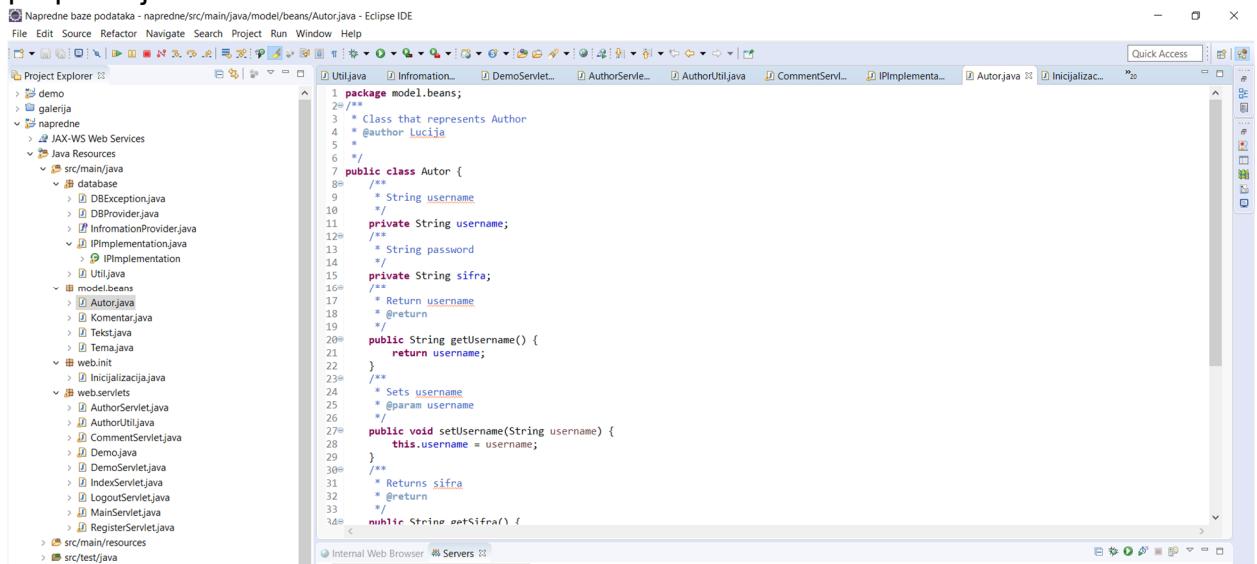
Sve web stranice same aplikacije se nalaze u posebnoj datoteci. Ostali kod je organiziran u 4 paketa, od čega se:

- **jedan (*database*) bavi isključivo bazama podataka**



```
22  /*
23  */
24  public class IPImplementation implements InfomrationProvider{
25      /**
26      * @inheritDoc
27      */
28      public List<Tekst> getAllTextsByAuthorUsername(String username) throws DBException, UnknownHostException {
29          DBCollection collection = Util.getCollection("Test", "tekstovi");
30          BasicDBObject whereQuery = new BasicDBObject();
31          whereQuery.put("autor", username);
32          DBCursor cursor = collection.find(whereQuery);
33
34          List<Tekst> tekstovi = new ArrayList<Tekst>();
35
36          while(cursor.hasNext()) {
37              BasicDBObject t = (BasicDBObject) cursor.next();
38
39              List<Komentar> listKomentara = getAllCommentsByTextId((int)t.getInt("text_id"));
40              List<Tema> listTema = getAllThemesByTextId((int)t.getInt("text_id"));
41              List<String> listSlike = getAllPicturesByTextId((int)t.getInt("text_id"));
42              Tekst tmpT = Util.makeText((int)t.getInt("text_id"), (String)t.getString("naslov"),
43                                         (String)t.getString("tekst"), username, listKomentara, listTema, listSlike);
44
45              tekstovi.add(tmpT);
46          }
47
48          Util.close();
49          return tekstovi;
50      }
51
52  }
53
54  /**
55  * @inheritDoc
56  */
```

- **drugi (*model.beans*) drži objekte baze podataka u obliku kojeg Java prepoznaće**



```
1 package model.beans;
2 /**
3  * Class that represents Author
4  * @author lucija
5  *
6  */
7 public class Autor {
8     /**
9      * String username
10     */
11     private String username;
12     /**
13      * String password
14     */
15     private String sifra;
16     /**
17      * Return username
18      * @return
19      */
20     public String getUsername() {
21         return username;
22     }
23     /**
24      * Sets username
25      * @param username
26      */
27     public void setUsername(String username) {
28         this.username = username;
29     }
30     /**
31      * Returns sifra
32      * @return
33      */
34     public String getsifra() {
```

- treći (*web.init*) služi samo za trenutak pokretanja aplikacije

```

package web.init;
import javax.servlet.ServletContextEvent;
public class Inicijalizacija implements ServletContextListener {
    /**
     * Class implements {@link ServletContextListener}.
     * @author Lucija Valentic
     */
    @WebListener
    public void contextInitialized(ServletContextEvent sce) {
    }
    /**
     * {@inheritDoc}
     */
    public void contextDestroyed(ServletContextEvent sce) {
    }
}

```

- i u četvrtom paketu (*web.servlets*) su razredi, to jest servleti koji pripremaju podatke za stranice aplikacije.

```

@WebServlet("/servleti/main")
public class MainServlet extends HttpServlet {
    /**
     * {@inheritDoc}
     */
    private static final long serialVersionUID = 1L;
    /**
     * {@inheritDoc}
     */
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        List<Autor> u = DBProvider.getIPPProvider().getAllAuthors();
        if(u != null && !u.isEmpty()) {
            Map<String, String> users = new HashMap<String, String>();
            for(Autor user : u) {
                users.put(user.getUsername(), user.getUsername());
            }
            req.setAttribute("users", users);
        }
        req.getRequestDispatcher("/WEB-INF/pages/Main.jsp").forward(req, resp);
    }
    /**
     * {@inheritDoc}
     */
}

```

Također, prije navedeni servleti mogu komunicirati i sa stranicama i sa bazom podataka preko objekta koji je posebno napravljen za tu svrhu (koji se nalazi u paketu za baze podataka).

8. Korisničko sučelje (HTML, CSS)

Kod za prikaz samih web stranica, kao i njihova stilska obilježja, se nalazi u programu sa kodom, u folderu `\Aplikacija\src\main\webapp\WEB-INF\pages`.

Dolje u nastavku je primjer isječka koda iz `Register.jsp` koji definira izgled korisničkog sučelja stranice za registraciju korisnika:

```
<title>

<c:choose>
<c:when test="\${currentUserId == null }">
Anonymous
</c:when>
<c:otherwise>
\${currentUserName }
</c:otherwise>
</c:choose>

</title>
</head>
<body>
<div class="pravokutnik"></div>
<div class="header"><h1>Regisriraj se</h1></div>

<div class="sadrzaj">
<br>
<br>
<br>
<br>
<form action="register" method="post">

Username:<input type="text" name="username">
<div class="error">
<c:if test="\${errors.get('username')!=null }">
<br>
```

Promotrimo i stilski prikaz nekog objekta uz pomoć CSS-a u istoj datoteci, npr.

```
input[type="text"]
{
    font-size:16px;
    font-family: "Indie Flower";
    color: #a3a3a3;
}
```

8. Zaključak

Programiranje ove aplikacije je bilo novo i poučno iskustvo, budući da većina studenata nije bila upoznata sa programskim jezikom Java prije početka projekta.

Sama interakcija baze podataka i koda u Javi je bila relativno jednostavna za provedbu nakon jasnog definiranja potreba (tj. potrebnih upita) i zahtjeva na podatke.

Pri konstruiranju aplikacije je oblik modela velikim dijelom diktirala forma (tada još zamišljene) aplikacije, tako da je bilo interesantno promatrati „*top-down approach*“ - ovdje u smislu - od finalnog produkta do potrebnog prikladnog modela baze podataka.

Sam oblik baze je očito i diktirala sama činjenica da MongoDB kao program za interakciju s bazama podataka nije primarno prilagođen relacijskim bazama – pa smo shodno i tome završili samo sa dvije kolekcije (tri ako brojimo kolekciju nastalu sa *MapReduce* frameworkom iz jedne od prvotnih), puno manje u odnosu na broj mogućih kategorija podataka (*slike, komentari, teme, tekstovi, autori...*).

Iako je konačni izgled aplikacije sasvim bazičan, sa ispunjenom funkcionalnošću te mogućnostima pohrane novih podataka i manipulacije starih, te novim iskustvom sa jezikom Java i prvim korištenjem MongoDB van njega samoga, uz konstrukciju prve vlastite veće baze podataka, projekt izrade blog-platforme je bio vrijedno i zanimljivo iskustvo.

9. Izvori

- <https://moodle.srce.hr/2019-2020/>
- <https://docs.mongodb.com/manual/>
- <https://markozupanic.hr/blogovi/>
- <https://www.oracle.com/java/technologies/javase-downloads.html>
- <https://www.java.com/en/download/>
- <https://maven.apache.org/download.cgi>
- <https://www.mongodb.com/download-center>
- <https://tomcat.apache.org/download-80.cgi>
- <https://www.docker.com/>