Andra LUNGU, grupa 341C3
Alexandru-Petrisor PAJARCU, grupa 343C3

Proiect Baze de Date 2: O retea sociala pentru programatori
< Coders' Ranking>

Acest proiect isi propune reunirea mai multor programatori intr-o retea sociala. Totodata, le ofera posibilitatea de a se conecta mai usor la site-urile lor favorite(e.g. LinkedIn, GitHub, etc).

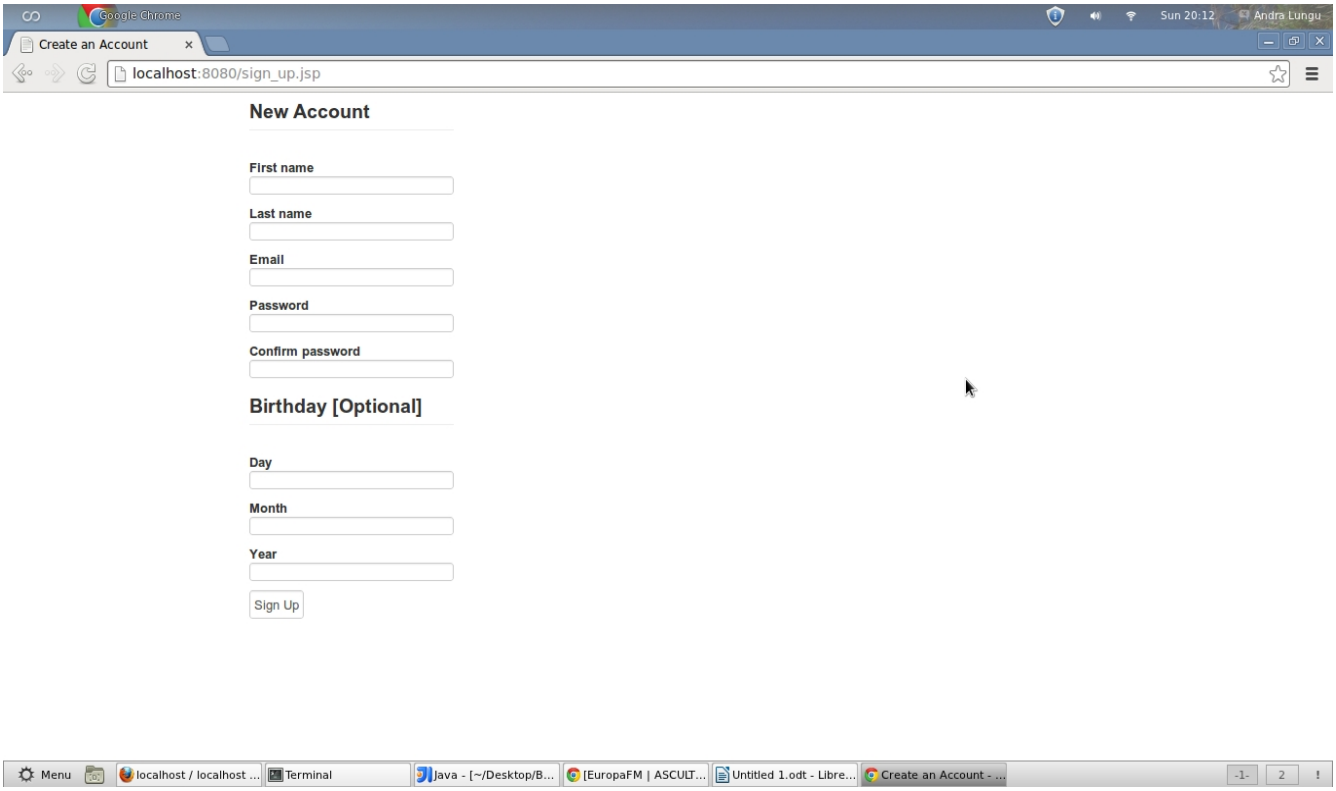
Pagina principala, accesibila dupa pornirea Apache Tomcat si a xampp-ului, are urmatorul design:

Aceasta pagina a fost conceputa folosind un JavaServletPage denumit index.jsp.
De aici un user poate efectua urmatoarele actiuni
→ accesarea paginii de login a site-urilor favorite: LinkedIn, StackOverflow, GitHub
→ logarea pe baza de username si parola
→ crearea unui cont de utilizator
→ cautarea explicatiile in limba romana a unor cuvinte englezesti

Conexiunea la baza de date se face folosind DriverManager . Codul care realizeaza conexiunea e urmatorul:

```
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost/coders_ranking", "root", "");
st = con.createStatement();
```

Formularul de Sign-Up se acceseaza dand click pe link-ul de Join Coders Ranking.



Folosindu-ne de sign_up.jsp preluam datele din textfield si le trimitem ca request de tip GET la new_user.jsp.

In acesta folosind cod JAVA si adaugam in baza de date un nou utilizator in tabela USER:

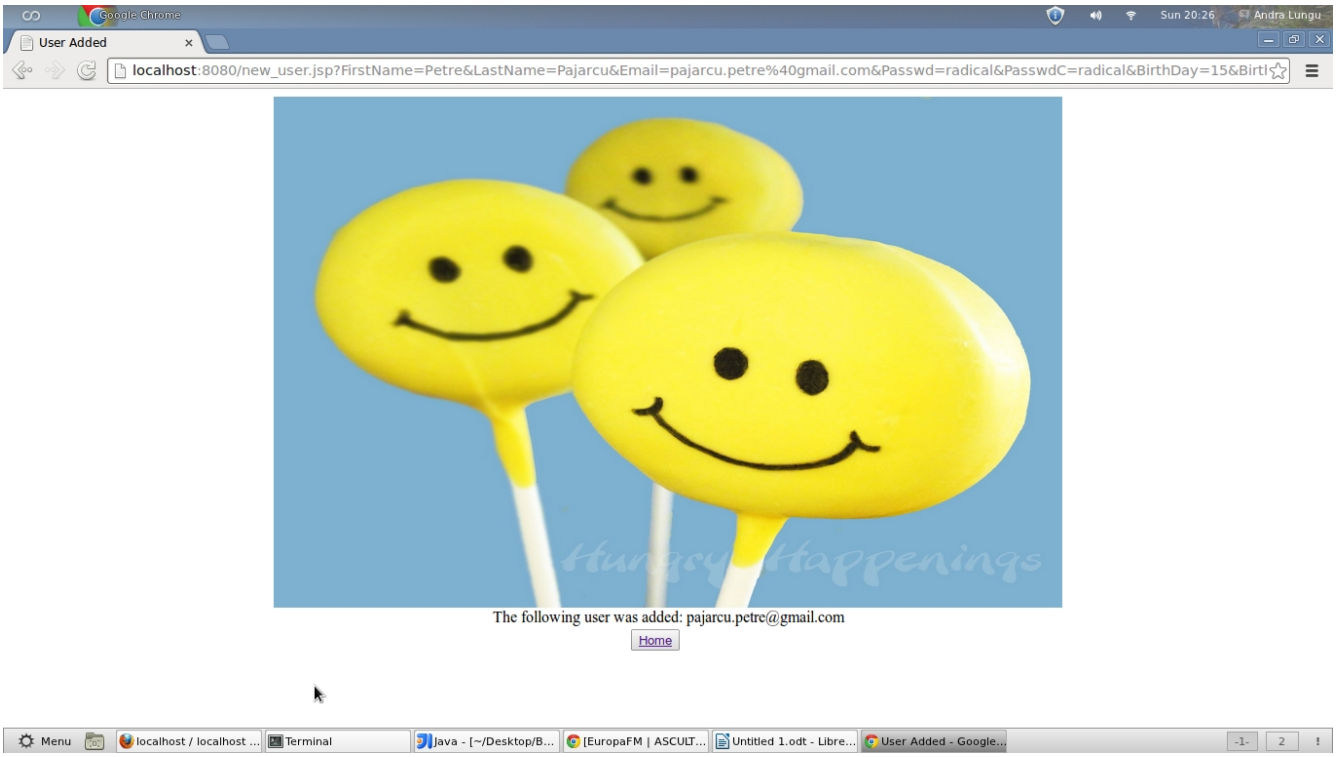
```
query = "INSERT INTO `user` VALUES(NULL, '"+ request.getParameter("FirstName")
+"','" + request.getParameter("LastName")+"','"
+ request.getParameter("Email") +"','" + request.getParameter("BirthDay")+
request.getParameter("BirthMonth")+ request.getParameter("BirthYear")+"','" +
password + "')";
st.executeUpdate(query);
```

Parola va fi criptata folosind algoritmul MD5.

In paralel se aduga in tabela USER_INFO o inregistrare pentru user curent cu valori default -1 si acelasi email din tabela USER.

Din considerente de identificare facila a utilizatorilor in tabela USER asignam id-ul 1 userului curent si id-ul cu numarul 2 userului pe a carui pagina de profil o vizualizam.

Dupa submiterea formului de sign-up, daca userul a fost adaugat cu success, se va afisa urmatoarea pagina. Utilizatorul va putea da click pe butonul Home pentru a reveni la pagina de login.



Dictionarul englez-roman se poate accesa dand click pe link-ul “Search for a word”. Pagina pe care userul o poate vizualiza are urmatorul design:

In baza de date au fost stocate perechi de valori (cuvant engleza, cuvant romana) .
In urma interogarii tabelei dictionar am aflat echivalentul in romana a cuvantului introdus de la tastatura.

Pentru logare, formularul are ca target o clasa Login de tip HttpServlet cu ajutorul careia se autentifica un user. Daca un user foloseste in posturile sau commenturile sale unul din cuvintele "banned", activitatea sa este suspendata, el este introdus intr-o tabela de "banned_users" si numai dupa expirarea unei cuante de timp, se mai poate loga pe profilul sau.

Pentru a putea fi recunoscut de server, servletul este declarat in fisierul web.xml astfel: (Similar procedam si pentru restul de fisiere HttpServlet utilizate)

```
 <servlet>
   <servlet-name>Login</servlet-name>
   <servlet-class>Simple.Login</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>Login</servlet-name>
   <url-pattern>/Login</url-pattern>
 </servlet-mapping>
```

Codul aferent clasei Login este urmatorul:

```
package Simple;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

public class Login extends HttpServlet {
   private static final long serialVersionUID = 1L;
   Statement st, st2, stBanned,st3;
   Connection con;
   String query;
   String parola;
   String username;
   ResultSet rs, rsBanned, rsTime,rs2;
   PreparedStatement ps;
   String sql = "SELECT CURTIME()";
   String crt;

   public Login() {
      super();
   }

   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
      try {
         Class.forName("com.mysql.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost/coders_ranking", "root",
"");
         st = con.createStatement();
         stBanned = con.createStatement();
         st2 = con.createStatement();
         st3 = con.createStatement();
         ps = con.prepareStatement(sql);
         rsTime = ps.executeQuery();

         username = request.getParameter("user");
         query = "SELECT * FROM `user` WHERE `email`='"+ username +"'"+" AND `id`>2";
         parola = (String) request.getParameter("password");
         parola = Encryption.crypt(parola);
         rs = st.executeQuery(query);

         query = "SELECT * FROM `banned_users`";
         rsBanned = stBanned.executeQuery(query);
         if (rs.next()) {
            while (rsBanned.next()) {
               if(username.equals(rsBanned.getString("email"))) {
                  while (rsTime.next()) {
                     crt = rsTime.getString(1);
                     query = "SELECT * FROM `banned_users` WHERE
(SUBTIME(CURTIME(),`time`)>'00:15:00' OR CURTIME()<`time`)" +
                           " AND `email`='"+username+"'";
                     rs2 = st3.executeQuery(query);
                     if (rs2.next()) {
                        query = "DELETE FROM `banned_users` WHERE `email` = '" + username
+"'";
                        st2.executeUpdate(query);
                        if (rs.getString("password_hash").equals(parola)) {
                           request.getRequestDispatcher("user_profile.jsp").forward(request,
response);
                           return;
                        }
                        else {
                           request.getRequestDispatcher("index.jsp").forward(request, response);
                           return;
                        }
                     }
                     else {
                        request.getRequestDispatcher("index.jsp").forward(request, response);
                        return;
                     }
                  }
               }
            }
```

```
            if (rs.getString("password_hash").equals(parola)) {
               request.getRequestDispatcher("user_profile.jsp").forward(request, response);
            }
            else {
               request.getRequestDispatcher("index.jsp").forward(request, response);
            }
         }
         else {
            request.getRequestDispatcher("index.jsp").forward(request, response);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      PrintWriter p=response.getWriter();
      p.println("Success!");
      doGet(request,response);
   }

}
```

Se observa implementarea binecunoscutelor metode GET si POST.

Practic, se cauta userul introdus in textfield dupa emailul sau. Se verifica daca acesta exista in baza de date, caz in care, daca nu este banned, se face redirectarea la profilul sau. Altfel, se revine la pagina principala. Daca userul este banned, dar i-a expirat perioada de suspendare, el este scos din tabela de "banned_users" si redirectat la pagina sa veche de profil.

In caz ca hasul parolei introduse nu coincide cu cel stocat in tabela "user", autentificarea este interzisa, userul fiind incurajat sa reintroduca usernameul si parola.

Frontendul listei posturilor si a commenturilor a fost realizat cu ajutorul jOuery mobile.

Odata autentificat, un user ajunge pe pagina sa de profil de unde poate efectua urmatoarele actiuni:

→ Sign Out – redirectare la pagina principala

→ alegerea unei imagini de profil (dintr-o lista predefinita)

→ cautarea customizata a unui user

→ cautarea unui user cu caracteristici comune cu userul curent

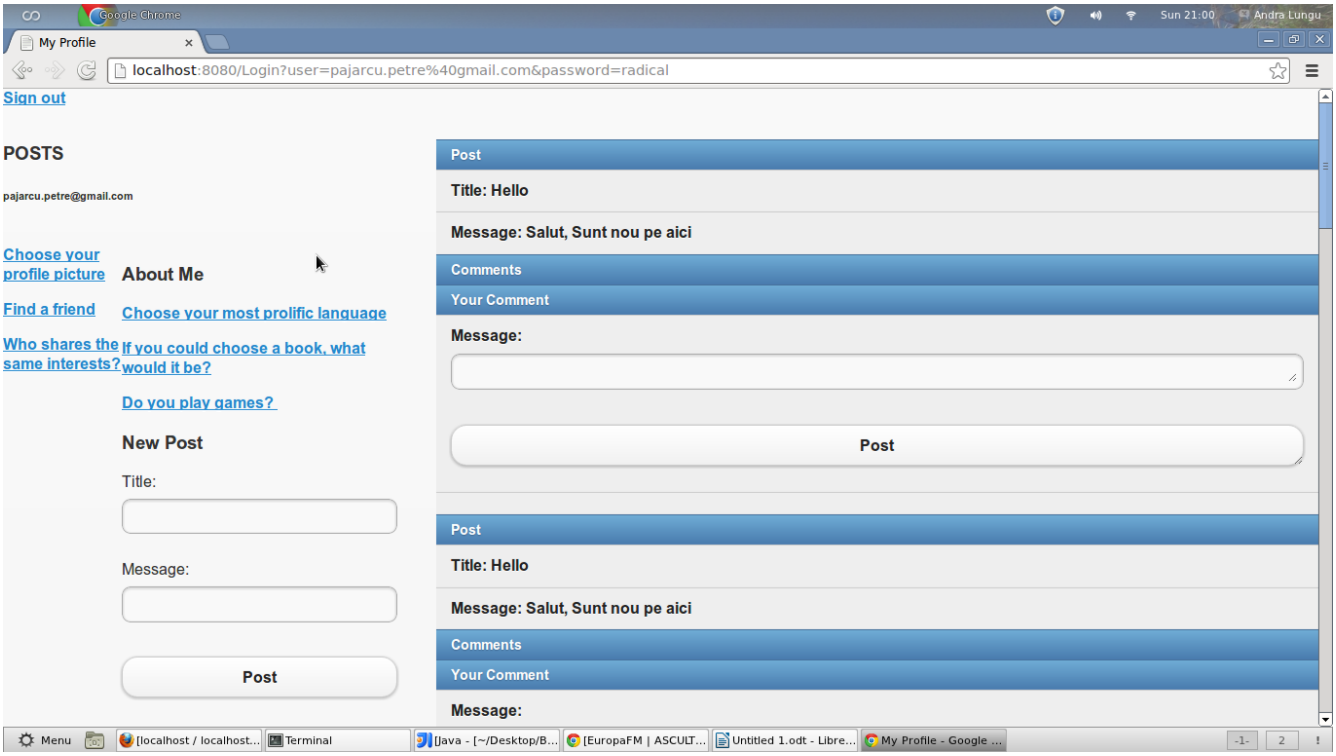
→ alegerea limbajului de programare favorit dintr-o lista (cu posibilitate de extindere a listei)

→ alegerea unei carti dintr-o lista (cu posibilitate de extindere a listei). Prietenii userului curent sunt incurajati sa cumpere aceasta carte in mod legal de pe site-ul amazon.com

→ alegerea jocului preferat dintr-o lista (de aceasta data) limitata de optiuni

→ postarea unor mesaje

→ comentarea mesajelor altor useri sau chiar a propriilor mesaje

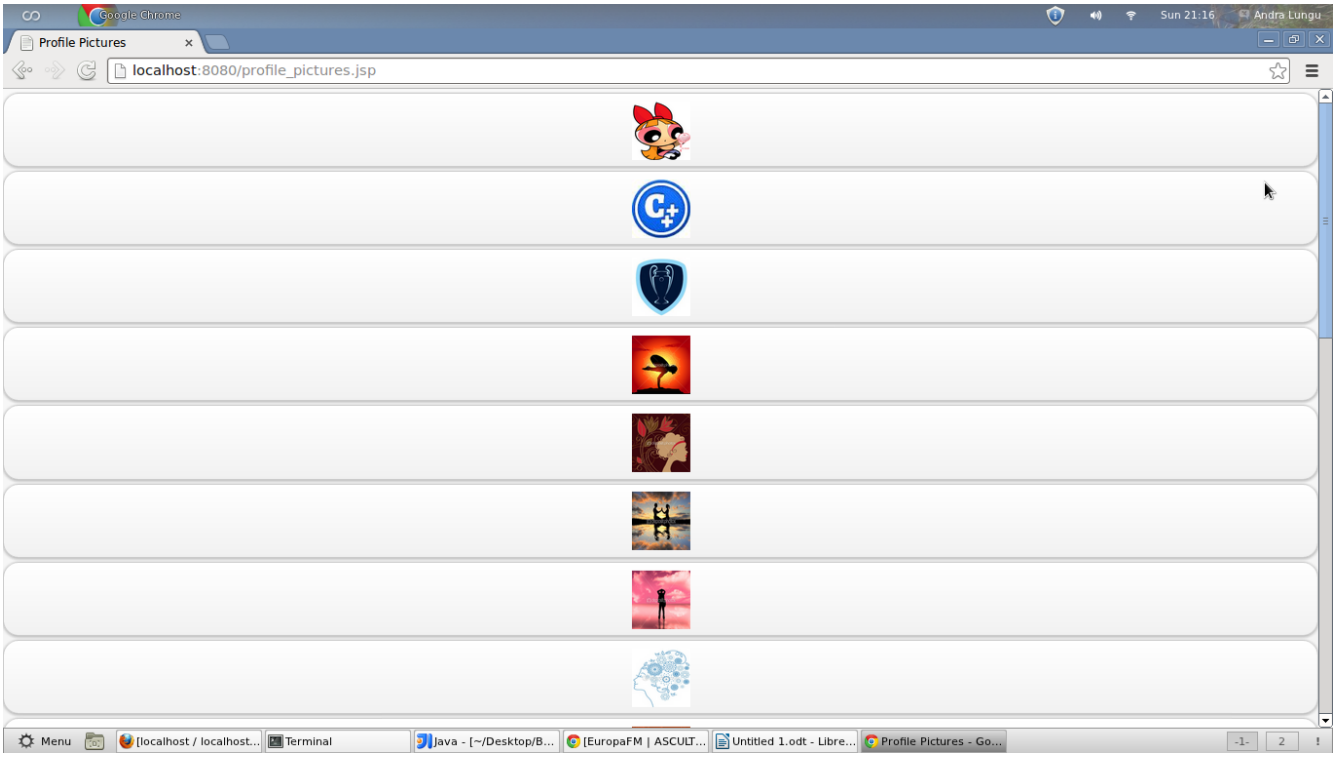


Formularul pentru adaugarea de mesaje in baza de date in tabela POSTS:

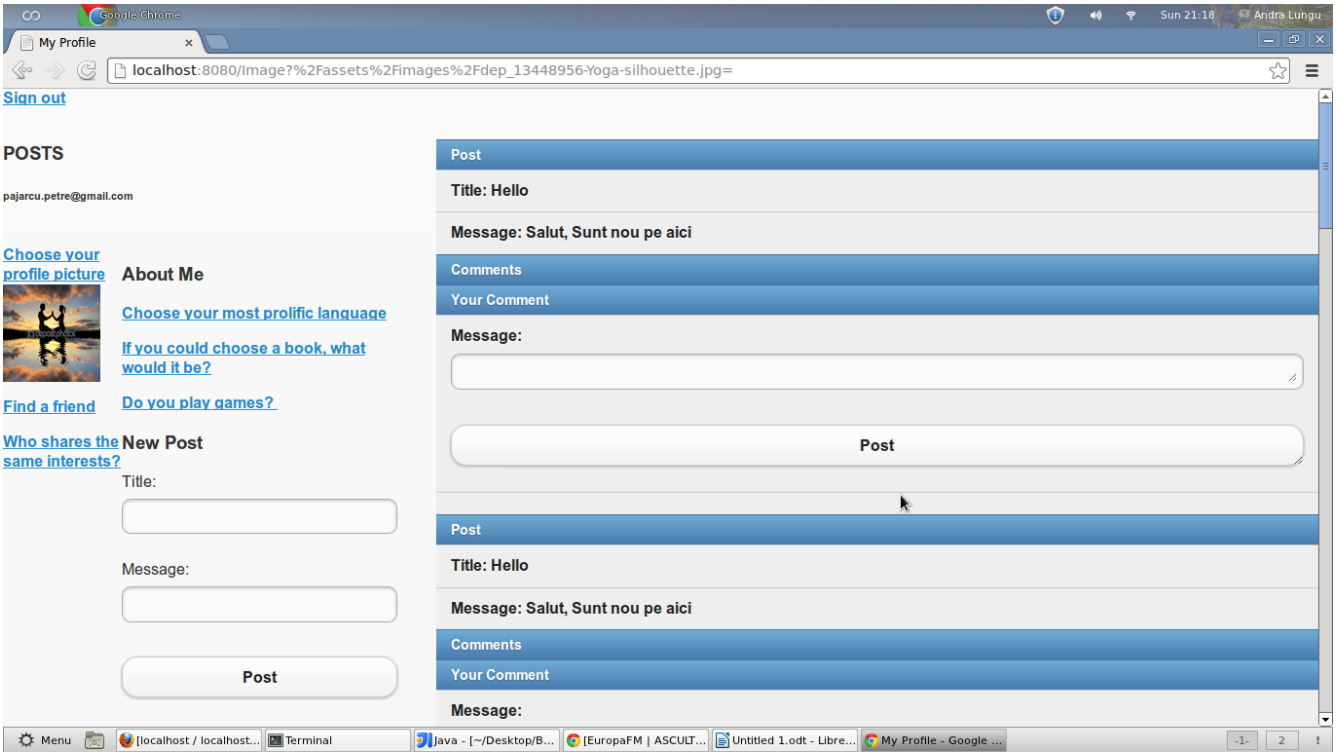
```
<%--Postarea unui nou mesaj--%>
<%-- Acesta e verificat de clasa Ban pentru a nu contine mesaje din clasa celor neautorizate--%>
<form action="Ban" method="GET">
Title: <input type="text" name="title" value= ${title} >
<br />
Message: <input type="text" name="message" value= ${message} >
<br />
<input type="submit" value="Post" />

<%
   /*adaugare mesaj in baza de date*/
   if(request.getParameter("title")!=null) {
      query = "INSERT INTO `posts` VALUES(NULL, '"+ currentUser +"','"+ request.getParameter("title") +"','" + request.getParameter("message")+"')";
      st.executeUpdate(query);
   }
%>
</form>
```

Prin apelarea clasei Ban la adaugare se verifica daca acel comentariu/post respecta normele de conduita impusa de site-ul nostru. Din nou se folosesc Servlet-uri pentru testarea de mesaj banned, cat si pentru comment-uri.

In cele ce urmeaza vom prezenta cum se schimba imaginea de profil a userului:



Dupa selectarea imaginii dorite se va schimba avatarul in cadrul profilului:

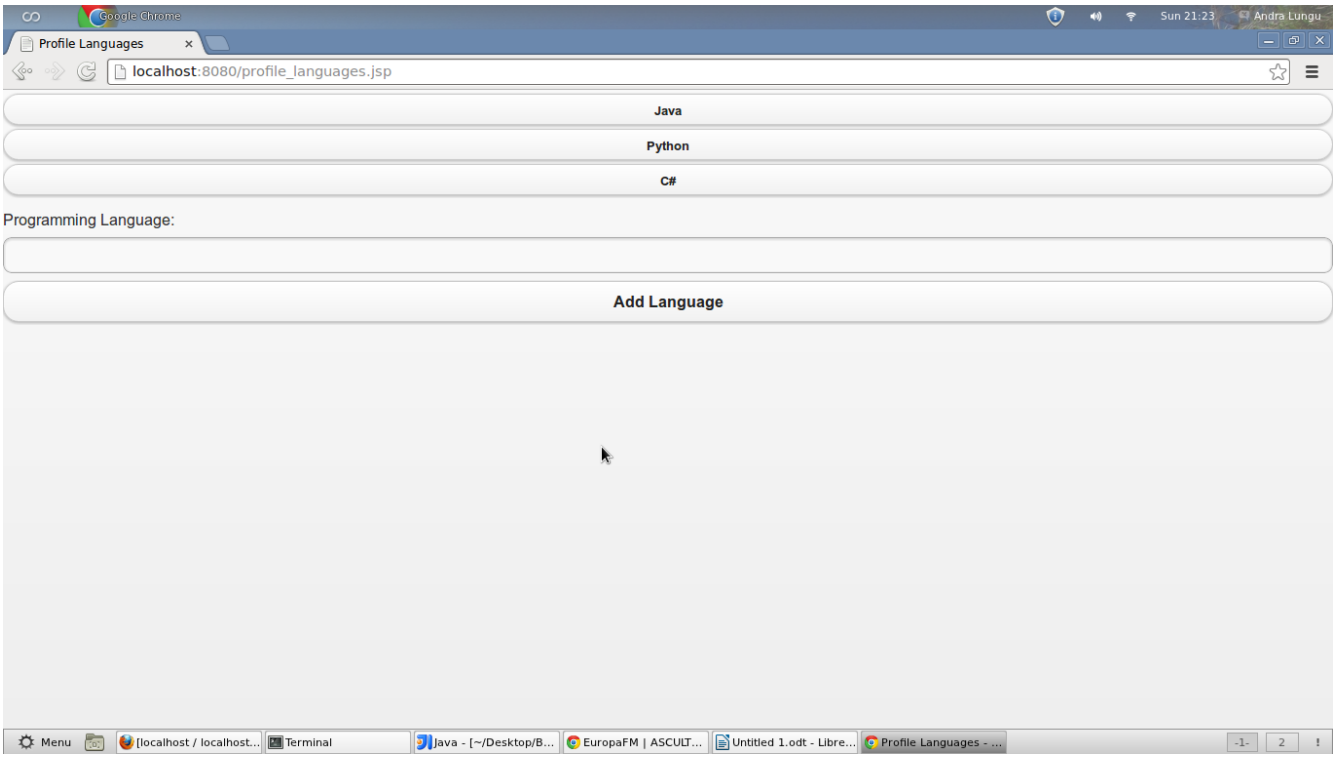


Codul aferent schimbarii imaginii se afla in servletul Image. Lista e realizata cu ajutorul profile_pictures.jsp.

Update-ul se realizeaza astfel:

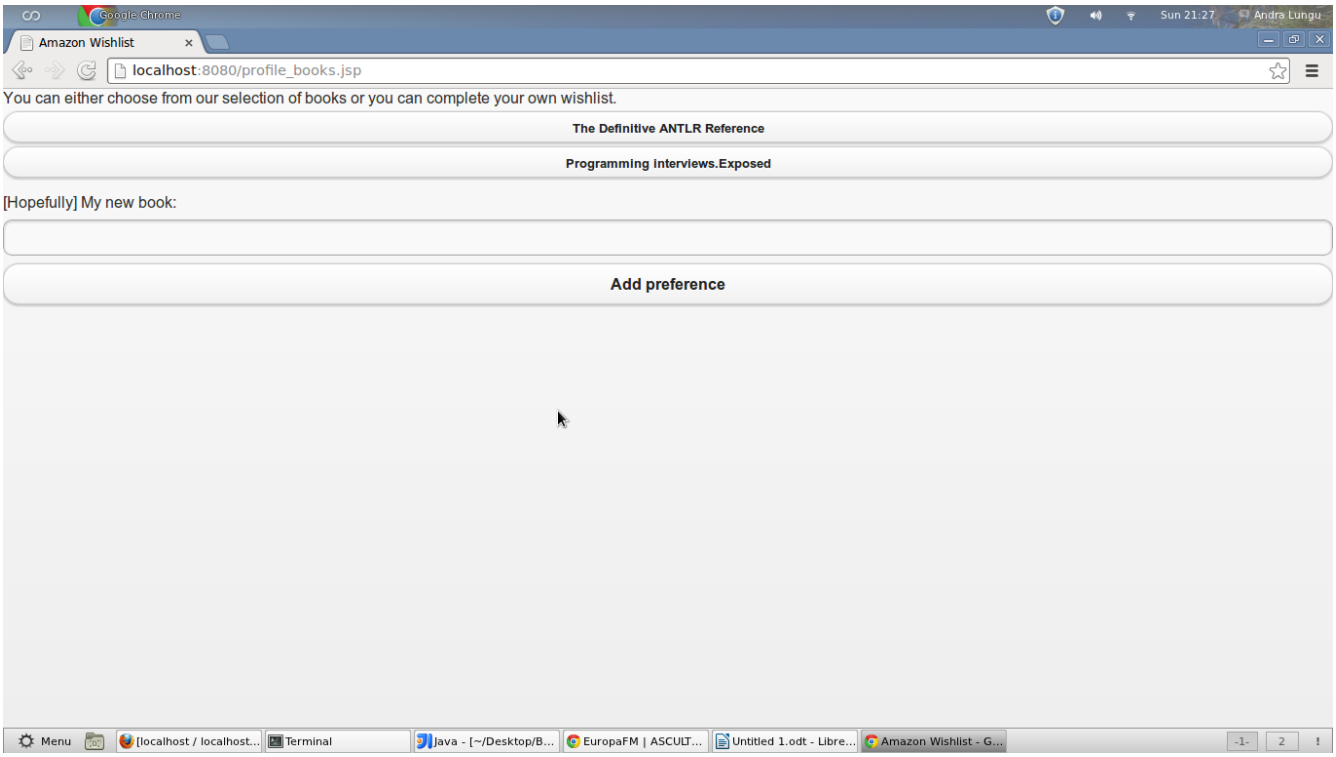
```
query = "SELECT * FROM `images` WHERE `location`='"+ location +"'";
 rs2 = st2.executeQuery(query);
 if(rs2.next()) {
    id_location = rs2.getInt("id");
 }
 else {
    query = "INSERT INTO `images` VALUES (NUll,'"+ location +"')";
    st3.executeUpdate(query);
    query = "SELECT * FROM `images` WHERE `location`='"+ location +"'";
    rs4 = st4.executeQuery(query);
    if(rs4.next()) {
       id_location = rs4.getInt("id");
    }
 }

 query = "UPDATE `user_info` SET `id_image`='"+ id_location +"' WHERE `email`='"+email+"'";
 st5.executeUpdate(query);
```

Limbajul se alege dand click pe “Choose your most prolific language” . Lista afisata este urmatoarea:



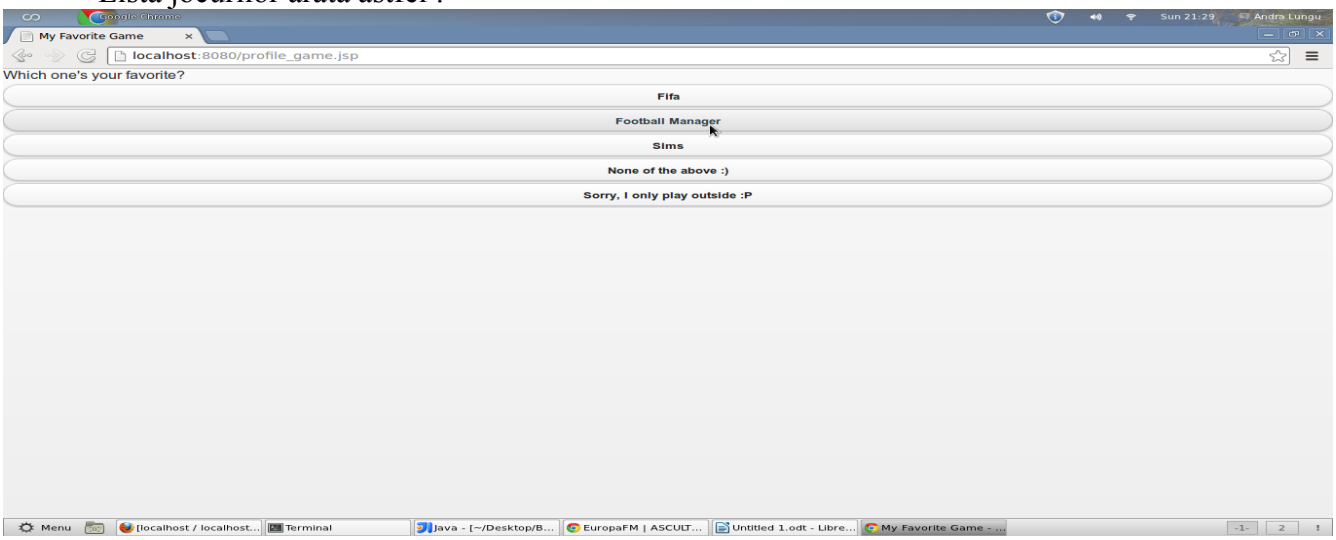
La aceasta se pot adauga si alte optiuni. Codul aferent este similar cu cel de la Image. La fel se foloseste un servlet pentru atingerea scopului dorit.

Alegerea cartii dorite se face dand click pe “If you could choose a book what would it be?”
Lista cartilor disponibile :

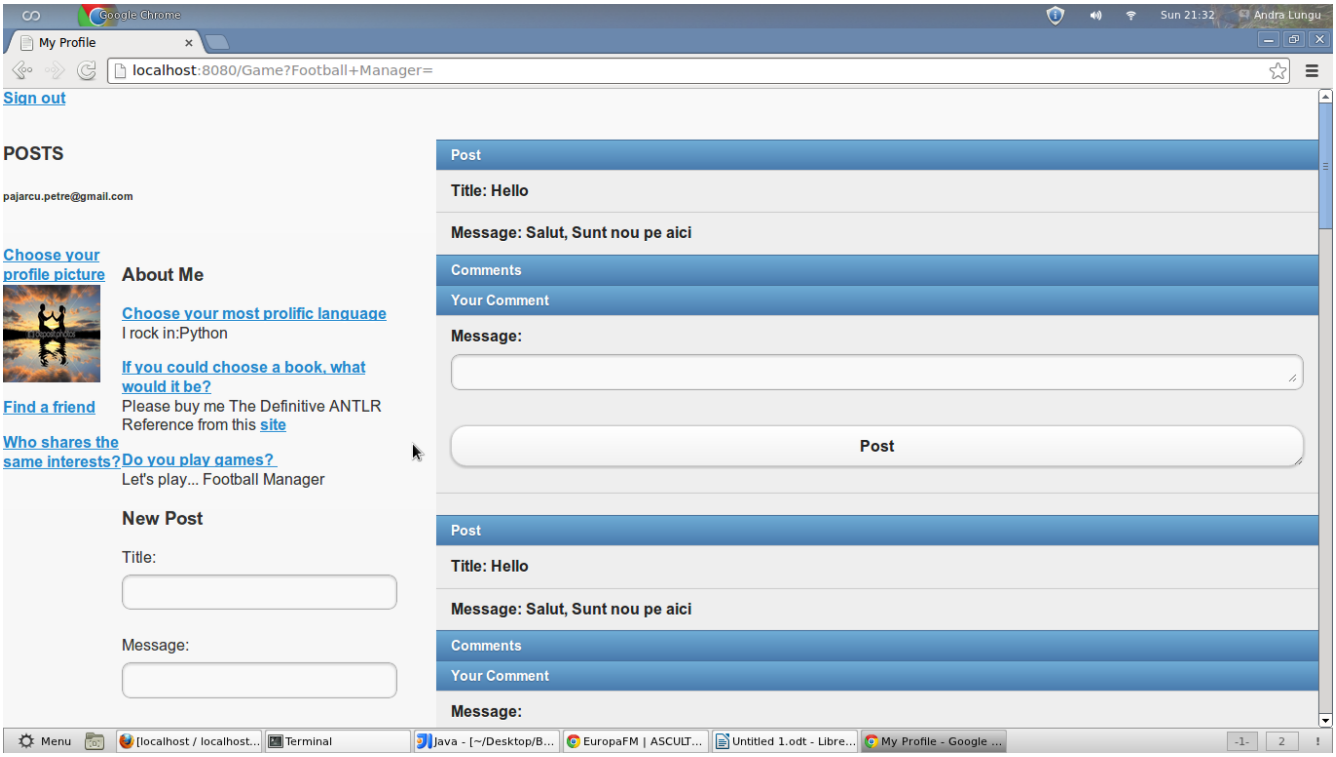


Lista poate fi ulterior imbogatita cu noi optiuni. Codul este similar cu cel de la Image.

Jocul preferat se alege cu “Do you play games”. De aceasta data nu se mai pot adauga noi optiuni.
Lista jocurilor arata astfel :



La final profilul va avea adaugate toate cele 3 optiuni:



Dand click pe link-ul “Find a friend” suntem redirectionati la alta pagina : find_friend.jsp . Aici vom putea cauta un user indicand anumite caracteristici pe care le dorim pentru prietenul respectiv. De mentionat ca acest raport extrage date din 5 tabele simultan.

Acest cod face extragerea din tabele:

Folosim pentru cautarea in tabele parametrii de request. In mod normal avem de cautat in cele 5 tabele folosind tabela de relationare USER_INFO.

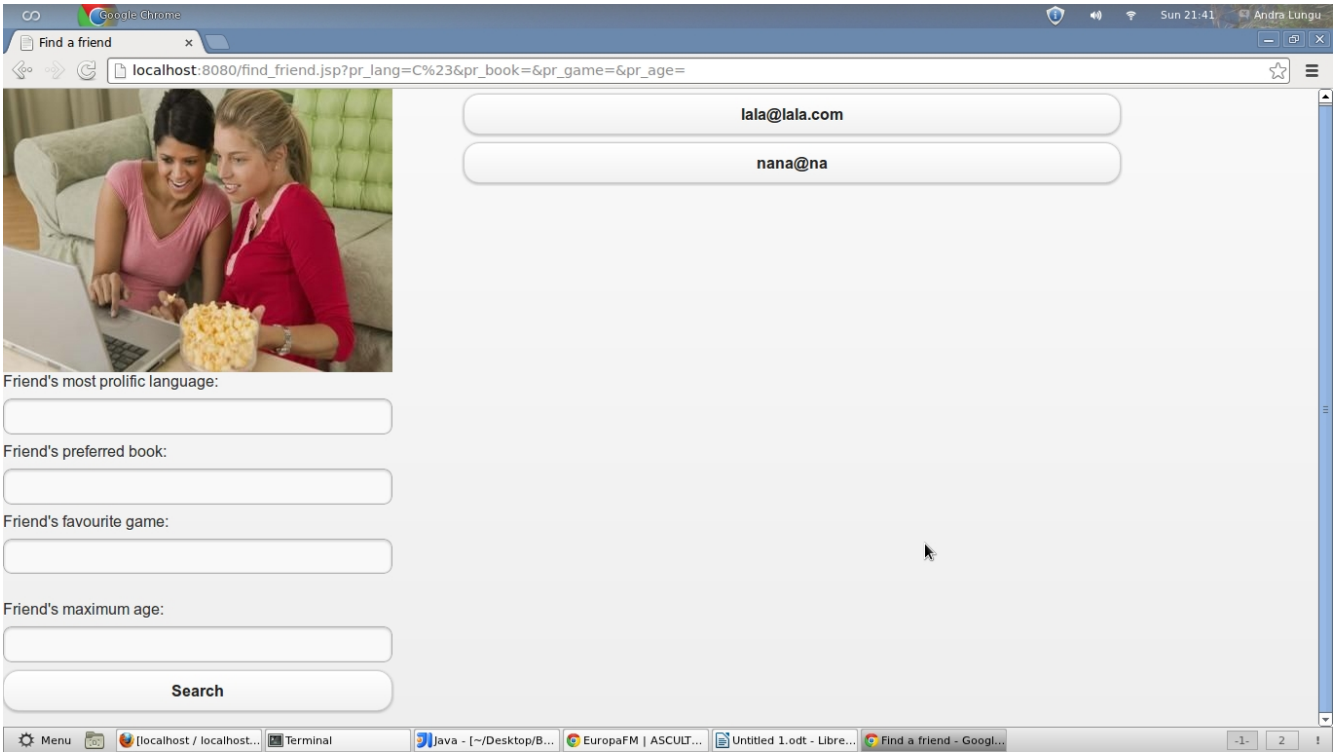
Daca unul sau mai multi parametri referitor la Limbaj, Varsta, Carti si Jocuri este completat acesta se va folosi pentru a face select din baza de date si mai restrictiv.

Vom folosi conjuctia intre operatia de baza si noile restrictii care trebuie satisfacute.

```
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost/coders_ranking", "root", "");
st = con.createStatement();

query ="SELECT * FROM `user`, `user_info`, `games`, `books`, `progr_languages` WHERE " +
    "`id_lang`=`progr_languages`.`id` AND `id_amazon`=`books`.`id` AND `id_game`=`games`.`id` AND" +
    "`user`.`email`=`user_info`.`email` AND `user`.`id`>2";
if (!request.getParameter("pr_lang").equals("")) {
   query += " AND `progr_languages`.`name`='"+request.getParameter("pr_lang")+
```

```
                """;
        }
        if (!request.getParameter("pr_book").equals("")) {
            query += " AND `books`.`name`='"+request.getParameter("pr_book")+
                """;
        }
        if (!request.getParameter("pr_game").equals("")) {
            query += " AND `games`.`name`='"+request.getParameter("pr_game")+
                """;
        }
        if (!request.getParameter("pr_age").equals("")) {
            query += " AND 2013-CONVERT(SUBSTRING(`birthday`,-4),UNSIGNED
INTEGER)<'"+request.getParameter("pr_age")+
                """;
        }
```

Textfield-urile se pot completa fie integral , fie partial selectia facandu-se in mod corespunzator.
Pagina de Find a Friend este urmatoarea:



Se poate observa deja rezultatul pentru introducerea valorii C# in primul camp.
Astfel am selectat toti userii care au ales C# ca limbaj de baza.

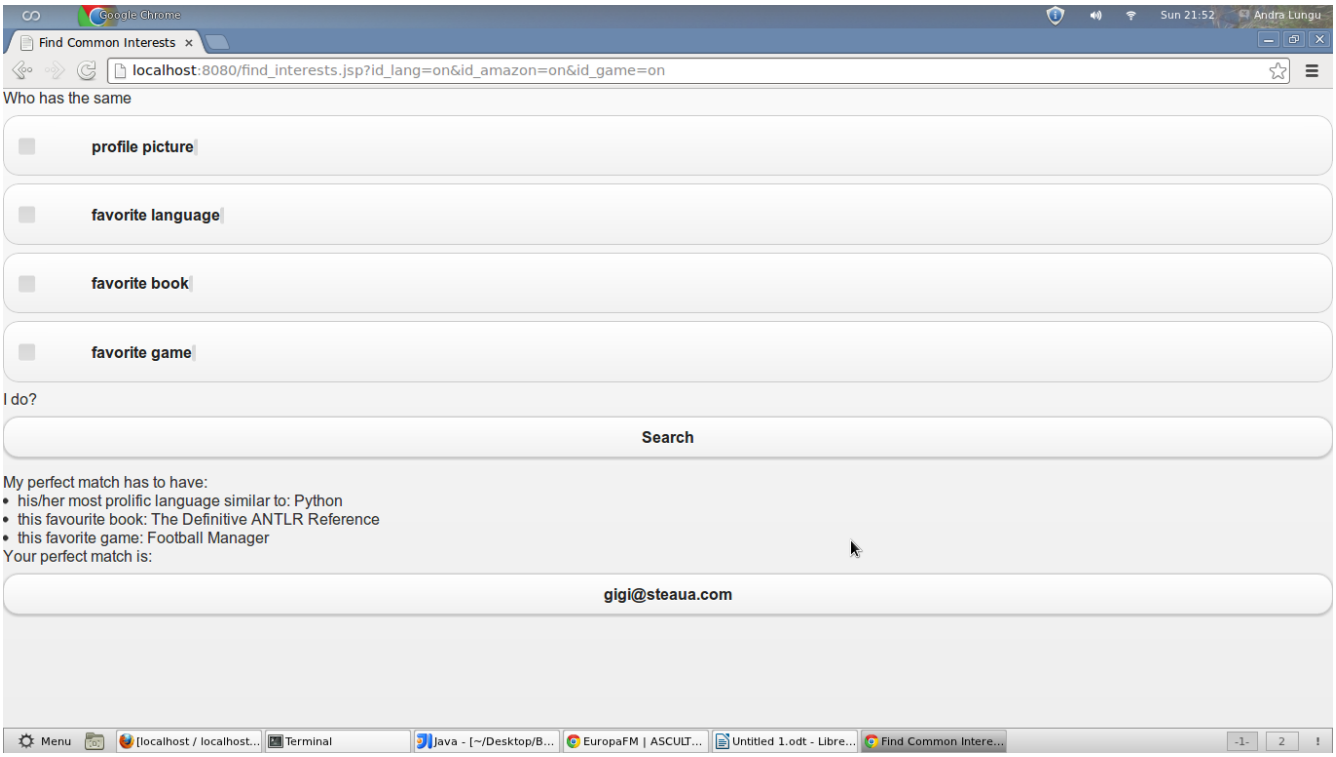
Daca nu completam nici un camp se vor printa toti userii care si-au completat campurile obligatorii.

Alta optiune posibila din profile este cea de a cauta prin bifarea unor checkboxuri a unor useri ce impart aceleasi interese cu userul curent.

Pagina prin care se realizeaza accesul la acele checkboxuri este usor accesibila prin clickul linkului “Who shares the same interests?”. Observam ca suntem redirectati pe pagina find_interests.jsp.

Dupa bifarea limbajului si a cartii preferate, se afiseaza un mesaj cu profilul dorit al prietenului. Profilul este introdus de cuvintele “My perfect match must have:” urmate de valorile campurilor selectate.

Chiar daca bifam toate campurile, userul ales nu este cel curent, este un alt user sau niciunul.



In acest caz, userul cu interese similare va fi gigi@steaua.com.

Si la cautarea prietenilor prin Find friend si la cea prin interese comune, profilul userului rezultat poate fi accesat prin clickul pe butonul cu adresa de email a acestuia.

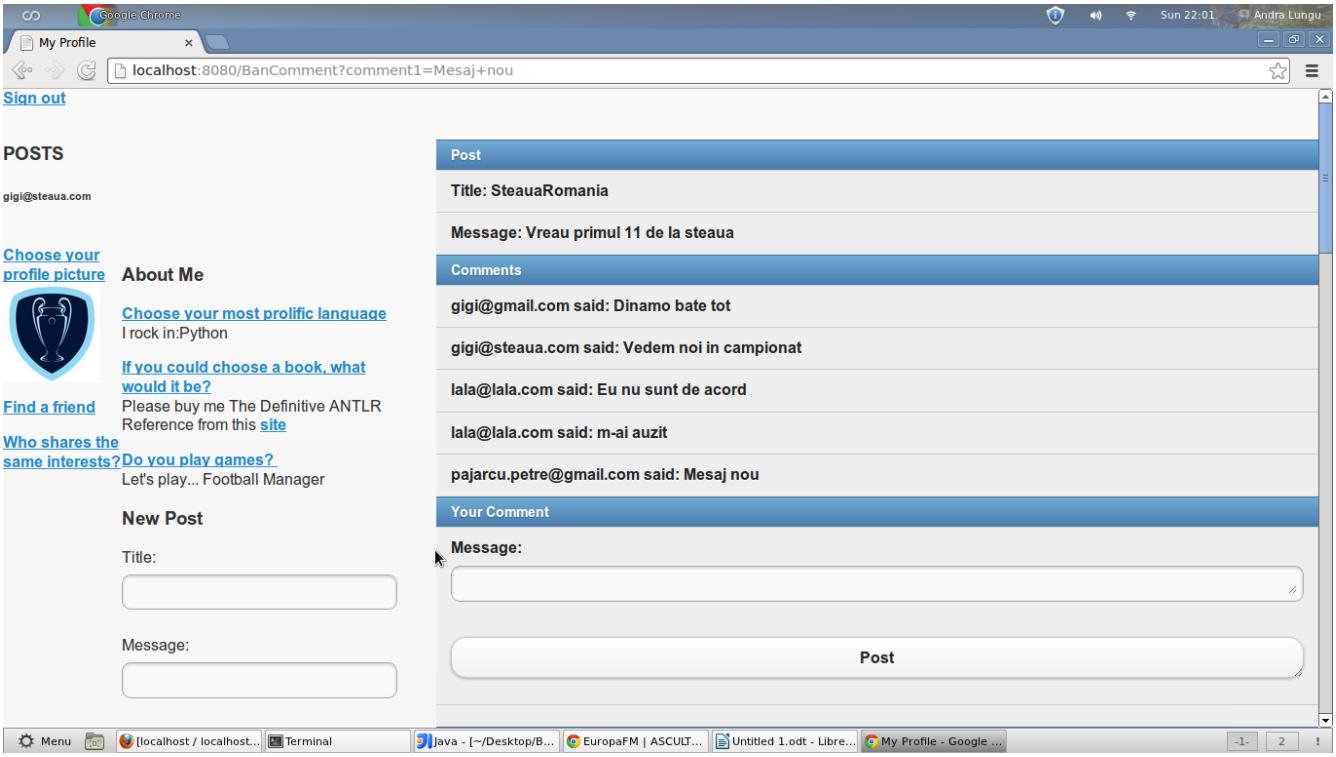
Dupa ce accesam profilul prietenului, vom putea posta mesaje. Atat mesajele cat si commenturile vor fi adnotate cu adresa de mail a expeditorului.

Este de mentionat ca si acest raport ca si cel anterior extrage date din mai mult de 4 tabele. Codul aferent acestei cautari este urmatorul:

```
if(rs.next() && (request.getParameter("id_lang")!=null||
request.getParameter("id_amazon")!=null
    ||request.getParameter("id_image")!=null||request.getParameter("id_game")!=null)) {
//          preiau emailul userului curent
currentUser = rs.getString("email");
%> My perfect match has to have: <br/> <%
query = "SELECT * FROM `user_info` WHERE `email`='"+currentUser+"'";
```

```
        rs2 = st2.executeQuery(query);
        if (rs2.next()) {
           query ="SELECT * FROM `user_info`, `games`, `books`, `progr_languages` , `images`
WHERE " +
                 "`id_lang`=`progr_languages`.`id` AND `id_amazon`=`books`.`id` AND
`id_game`=`games`.`id` AND" +
                 "`images`.`id`=`user_info`.`id_image` AND `user_info`.`id`!='"+rs2.getString("id")
+"'";
           if (request.getParameter("id_lang")!=null){
              query += "AND `user_info`.`id_lang`='"+rs2.getInt("id_lang")+"'";
              query2 ="SELECT * FROM `progr_languages` WHERE `id`='" +
rs2.getInt("id_lang") +"'";
              rs5 = st5.executeQuery(query2);
              if(rs5.next()) {
              %> <li> his/her most prolific language similar to: <%
out.write(rs5.getString("name"));%> </li> <%}
           }
           if (request.getParameter("id_image")!=null){
              query += "AND `user_info`.`id_image`='"+rs2.getInt("id_image")+"'";
              query2 ="SELECT * FROM `images` WHERE `id`='" + rs2.getInt("id_image")
+"'";
              rs4 = st4.executeQuery(query2);
              if(rs4.next()) {
              %><li> this image  <img src='<% out.print(rs4.getString("location")); %>'
style="height:100px; width:100px" /></li> <% }
           }
           if (request.getParameter("id_amazon")!=null){
              query += "AND `user_info`.`id_amazon`='"+rs2.getInt("id_amazon")+"'";
              query2 ="SELECT * FROM `books` WHERE `id`='" + rs2.getInt("id_amazon")
+"'";
              rs6 = st6.executeQuery(query2);
              if(rs6.next()) {
              %> <li> this favourite book: <% out.write(rs6.getString("name"));%></li><% }
           }
           if (request.getParameter("id_game")!=null){
              query += "AND `user_info`.`id_game`='"+rs2.getInt("id_game")+"'";
              query2 ="SELECT * FROM `games` WHERE `id`='" + rs2.getInt("id_game") +"'";
              rs7 = st7.executeQuery(query2);
              if(rs7.next()) {
                 %> <li>this favorite game: <% out.write(rs7.getString("name"));%></li><% }
           }
        }
     }
     rs3 = st3.executeQuery(query); %>
```

Dupa ce userul curent intra pe profilul altui user poate posta comentarii



Astfel am putut realiza o aplicatie web care foloseste ca mecanism de interactiune cu baza de data JDBC, ca limbaj pentru scrierea query-urilor MySQL, ca limbaj de scriere a aplicatiei Java, HTML, CSS, jQuery mobile.

Pentru a putea rula aplicatia, un utilizator va trebui sa aiba pachetul mysql-connector-java-5.1.13-bin.jar, axis, tomcat 7 si o versiune stabila de JDK.

Ca IDE pentru dezvoltare a aplicatiei s-a folosit IntelliJ IDEA Ultimate versiunea 12. Cu optiunea de Application Servers.

Am salvat datele de test intr-un fisier sql prezent in CD-ul atasat documentatiei.