



**Universitatea POLITEHNICA din București**

Facultatea de electronică, Telecomunicații și Tehnologia Informației



# **PROGRAMAREA INTERFETELOR PENTRU BAZA DE DATE**

Profesor coordonator: Ș. I. Dr. Ing. Pupezescu Valentin

STUDENT: CANDEA SEBASTIAN CONSTANTIN  
GRUPA: 431Aa

# CUPRINS

1) Tema proiectului.....	pag. 3
2) Descrierea sistemului de gestiune a bazelor de date MySQL.....	pag.3
3) Tehnologia JSP utilizata in dezvoltarea aplicatiei.....	pag.4
4) Limbajul HTML si utilitatea sa in aplicatii.....	pag. 4
5) Descrierea Aplicatiei.....	pag. 5
6) Concluzii.....	pag.16
7) Bibliografie.....	pag.16

## 1) Tema proiectului

Tema proiectului constă în dezvoltarea unei aplicații ce conține o baza de date, creată în sistemul de gestionare a bazelor de date MySQL.

Interfețele vor trebui să permită utilizatorului să execute următoarele operații pe toate tabele: vizualizare, adăugare, modificare și ștergere de date. Vizualizarea tabelelor de legătură va presupune vizualizarea datelor referite din celelalte tabele.

Pentru tema individuală am ales tehnologia JSP. Asocierea pentru tabela din baza de date este de tipul M:N.

## 2) Descrierea sistemului de gestiune a bazelor de date MySQL

### - Ce este MySQL?

MySQL este un sistem de gestionare a bazelor de date relationale open source care este utilizat în principal pentru aplicațiile online. MySQL poate crea și gestiona baze de date foarte utile (cum ar fi informații despre angajați, inventar și multe altele), la fel ca alte sisteme, cum ar fi popularul Microsoft Access. În timp ce Microsoft Access, MySQL și alte sisteme de gestionare a bazelor de date servesc scopuri similare (de a găzdui datele), utilizarea diferă foarte mult.[1]

MySQL este componenta integrată a platformelor LAMP sau WAMP (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului care este adesea combinat cu MySQL și denumit Duo-ul Dinamic. În multe cărți de specialitate este precizat faptul că MySQL este mult mai ușor de învățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”. [2]

### - Cum administrează MySQL bazele de date?

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau, prin descărcare de pe internet, o interfață grafică: MySQL Administrator și MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, phpMyAdmin.[2]

### - De ce este atât de util MySQL?

Baza de date MySQL este folosită în principal ca mijloc de a stoca date pentru aplicații mari, bazate pe web. Site-uri precum WordPress, iStock, GitHub, Facebook, NASA, Marina SUA, Tesla, Scholastic, Spotify, YouTube, Netflix, Glasses Direct, Symantec (și multe altele)

folosesc baza de date MySQL ca mijloc de stocare a datelor pe din interiorul sau exteriorul site-urilor web și serviciilor interne. [1]

### 3) Tehnologia JSP utilizata in dezvoltarea aplicatiei

#### - Ce este JSP?

Java Server Pages este o simpla dar puternica tehnologie folosita pe partea de server pentru a genera continut HTML dinamic. JSP este o extensie directa a Java Servlets si furnizeaza o modalitate de a separa partea de procesare de cea de prezentare. Motorul JSP este doar un alt Servlet, mapat la extensia \*.jsp. [5]

Paginile JSP sunt create sa suporte mai multe tipuri de documente structurate, indeosebi HTML si XML. În general, JSP-urile folosesc anumite informatii pe care le trimit la server intr-o cerere HTTP care interactioneaza cu datele existente pe acesta si creaza dinamic un raspuns organizat intr-un format standard (HTML, DHTML, XML, etc.) sau intr-un format text sau neorganizat ce va fi trimis înapoi clientului. [2]

#### - Ce este SERVLET?

Un servlet este un fișier java care poate solicita clientul, proceseaza-l și furnizează un fișier HTML ca raspuns. Exista mai multe servlete in interiorul containerului web. De asemenea, este posibila maparea mai multor cereri la un servlet. Prin urmare, toate aceste configuratii sunt incluse in acest fisier special, care este fisierul web.xml.[5]

#### - Care este diferenta majora intre JSP si SERVLET?

Principala diferenta între JSP și Servlet este ca JSP este un limbaj de scripting al paginii web care poate genera continut web dinamic în timp ce servletul este un program Java care este deja compilat și folosit pentru a crea conținut dinamic de web. Intr-o aplicatie web obisnuita, clientul solicita o pagina Web de la server și serverul raspunde inapoi cu pagina necesara. Aceste pagini pot fi fie statice, fie dinamice. Continutul unei pagini statice este deja creat. Continutul unei pagini dinamice este creat in timpul rularii. JSP și Servlet sunt două metode în Java pentru a crea pagini web dinamice. JSP este tradus si compilat intr-un servlet de catre containerul web. Pe de alta parte, un servlet este un program Java care este gestionat de containerul web.[5]

### 4) Limbajul HTML si utilitatea sa in aplicatii

#### - Ce este HTML?

Unul din primele elemente fundamentale ale WWW ( World Wide Web ) este HTML ( Hypertext Markup Language ), care descrie formatul primar în care documentele sunt distribuite și văzute pe Web. Multe din trasaturile lui, cum ar fi independenta fata de platforma, structurarea formătării și legaturile hypertext, fac din el un foarte bun format pentru documentele Internet și Web.[4]

- In ce scop utilizam limbajul HTML?

Scopul HTML este mai degraba prezentarea informatiilor – paragrafe, fonturi, tabele ș.a.m.d. decat descrierea semanticii documentului. In cadrul dezvoltarii web de tip front-end, HTML este utilizat impreuna cu CSS și JavaScript.[2]

## 5) Descrierea Aplicatiei

- Baza de date

Tema individuala se bazeaza pe crearea unei baze de date ce are 2 tabele in asociere M:N. Tabelele sunt: autori si proiecte. Pentru cele 2 tabele am ales câteva attribute caracteristice:

Pentru tabela autori: am ales ca si cheie primara idautori, restul atributelor sunt nume, prenume, adresa, varsta.

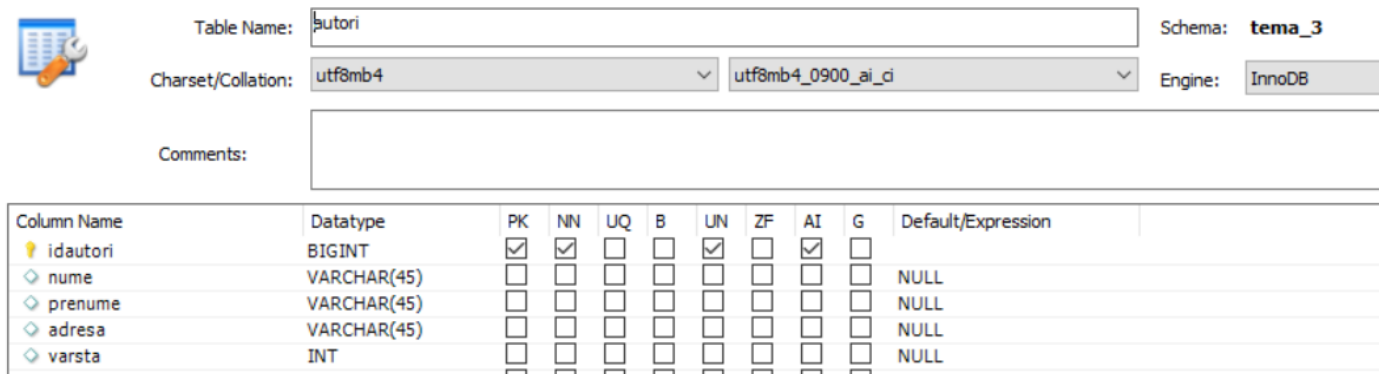







Table Name:  Schema: **tema\_3**

Charset/Collation:   Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idautori	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 nume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 prenume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 adresa	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 varsta	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Pentru tabela proiecte: am ales ca si cheie primara idproiecte, restul atributelor sunt: denumire, tematica, termen\_limita.

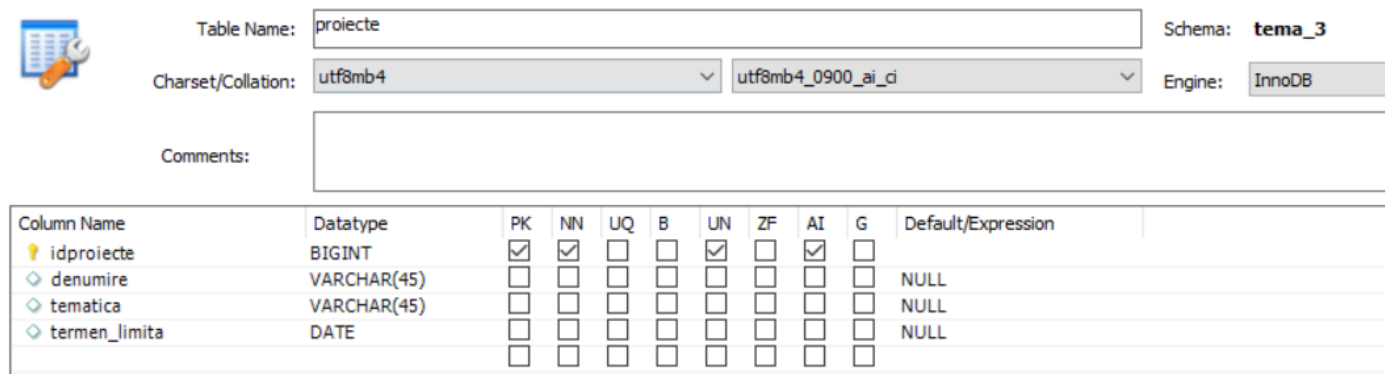






Table Name:  Schema: **tema\_3**

Charset/Collation:   Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idproiecte	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 denumire	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 tematica	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 termen_limita	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

- Ce este asocierea M:N?

Asocierea M:N (mai-multi-la-mai-multi) are ca si caracteristica faptul ca fiecarui element inregistrat intr-o tabela ii pot fi asociate mai multe elemente din cealalta tabela si invers. [3]

De exemplu in cazul nostru, un autor poate fi asociat mai multor proiecte, asa cum si un proiect poate fi asociat mai multor autori.

- Ce reprezinta tabela de legatura?

Pentru a crea o relație mai-multi-la-mai-multi, trebuie sa se creeze o a treia tabela denumita deseori tabela de joctiune, care imparte relatia mai-multi-la-mai-multi în doua relatii unu-la-mai-multi. In cazul nostru, am ales ca si tabela de legatura tabela asociere. In aceasta noua tabel, attributele ce au fost selectate ca si chei primare pentru tabelele anterioare vor deveni chei straine (FK) pentru tabela de legatura asociere.

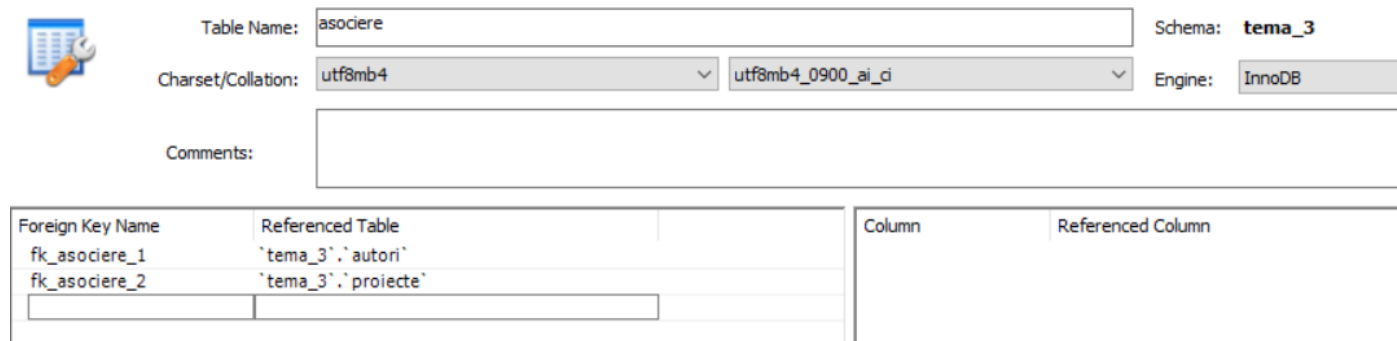


Table Name:  Schema: **tema\_3**

Charset/Collation:   Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
fk_asociere_1	'tema_3', 'autori'		
fk_asociere_2	'tema_3', 'proiecte'		

Pentru aceasta tabela am ales „idasociere” ca si cheie primara. Restul atributelor sunt cheile straine idproiecte si idautori.

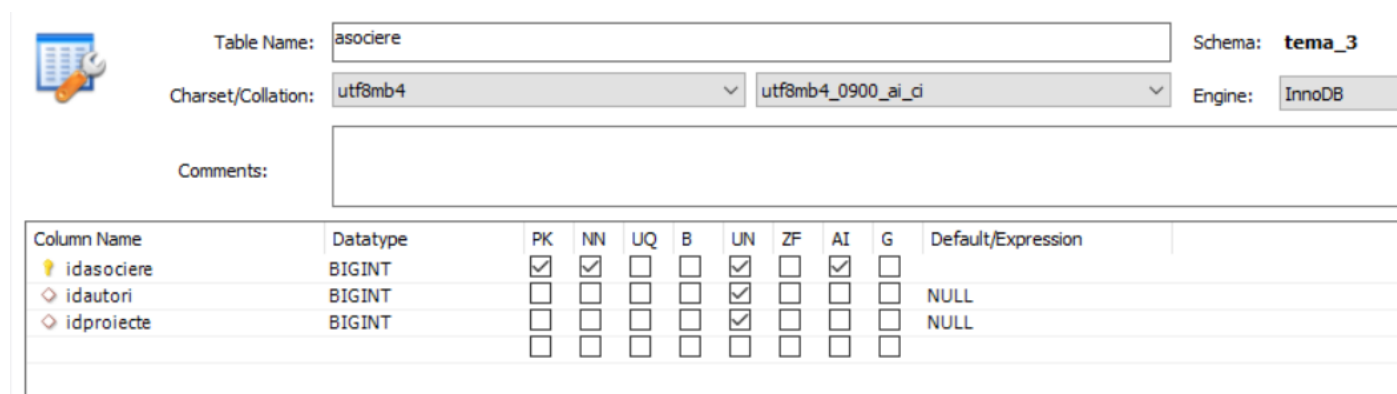


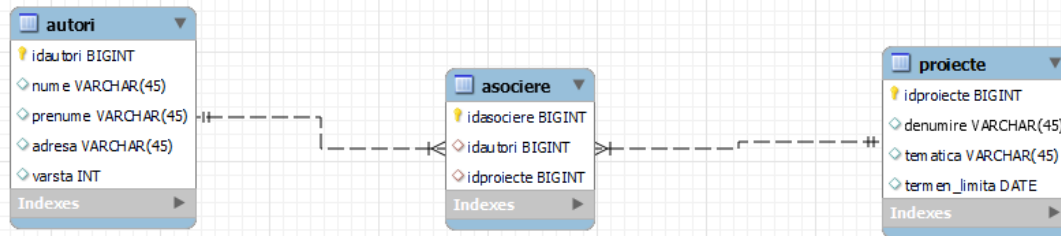
Table Name:  Schema: **tema\_3**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idasociere	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
idautori	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
idproiecte	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

## DIAGRAMA LOGICA A BAZEI DE DATE



### De ce este importanta diagrama logica a bazei de date?

Aceasta ofera o reprezentare logica detaliata a datelor celor 3 tabele, a relatiilor dintre ele.

### Ce relatii exista intre cele trei tabele?

- Intre autori si asociere exista o asociere de tipul 1:N
- Intre autori si proiecte este de tipul M:N
- Intre proiecte si asociere este de tipul 1:N

## FUNCTIONALITATEA APLICATIEI

### a) Arhitectura proiectului

Proiectul realizat in tehnologia JSP are urmatoarea structura:

- Un pachet DB ce continut clasa JavaBean, ce are rolul de a oferi toate functionalitatile principale ale interfetei bazei de date si anume: conectica, operatiile de afisare, adaugare, modificare, stergere.
- Folder-ul webapp ce continut toate paginile JSP, ce au rolul de a importa functionalitatile din clasa JavaBean, fiind conectate intre ele si implicit conectate la pachetul DB, ce realizeaza partea dinamica a proiectului.

### b) Implementarea functiilor

Toate paginile JSP contin structuri de cod java, ce ofera aplicatiei comportamentul dinamic. Ca sa utilizam functiile din clasa java, trebuie sa utilizam tag-urile JSP.

Un exemplu putem regasi in pagina JSP tabela\_autori:

```
7      <title>Tabela Autori</title>
8  </head>
9  <jsp:useBean id="jb" scope="session" class="db.JavaBean" />
10 <jsp:setProperty name="jb" property="*" />
11 <body>
12
13     <h1 align="center"> Tabela Autori:</h1>
```

Jb este un obiect instanta la clasa JavaBean, astfel facandu-se legatura directa intre paginile JSP si functiile pe care dorim sa le implementam.

Cum se realizeaza conexiunea?

```
27 <%
28     jb.connect();
29     ResultSet rs = jb.vedeTabela("autori");
30     long x;
31     while (rs.next()) {
32         x = rs.getInt("idautori");
33     }
34 %>
```



In tabela pe care am dat-o exemplu, am creat un formular, in cadrul caruia prin instructiunea `jb.connect()` ne legam la functia `connect()` din clasa `JavaBean`:

```
public void connect() throws ClassNotFoundException, SQLException, Exception {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/tema_3?useSSL=false", "root", "123456");
    } catch (ClassNotFoundException cnfe) {
        error = "ClassNotFoundException: Nu s-a gasit driverul bazei de date.";
        throw new ClassNotFoundException(error);
    } catch (SQLException cnfe) {
        error = "SQLException: Nu se poate conecta la baza de date.";
        throw new SQLException(error);
    } catch (Exception e) {
        error = "Exception: A aparut o exceptie neprevazuta in timp ce se stabilea legatura la baza de date.";
        throw new Exception(error);
    }
} // connect()
```

Obiectul “con” este cel prin intermediul caruia se realizeaza toate operatiile pe baza de date. In functia `connect()` : se incarca driver-ul de MySQL.

Prin functia `getConnection(...)` apelam functia de conectare si are ca attribute baza de date realizata in MySQL, user-ul si parola utilizatorului.

Totodata, exista si o functie `disconnect()` ce realizeaza `close()` pe obiectul de conexiune con:

```
64
65 public void disconnect() throws SQLException {
66     try {
67         if (con != null) {
68             con.close();
69         }
70     } catch (SQLException sqle) {
71         error = ("SQLException: Nu se poate inchide conexiunea la baza de date.");
72         throw new SQLException(error);
73     }
74 } // disconnect()
75
```

Cum se afiseaza dartaile?

In primul rand alegem ca si exemplu tabela albume. Functia din clasa `JavaBean` este `vedeTabela()` ce primeste ca si atribut numele tabelii pe care dorim sa facem interogarea si returneaza un set de rezultate: linii de tip “autori”, ce le implementeaza intr-un obiect de tip `results`. Acesta este declarat initial null.

```
public ResultSet vedeTabela(String tabel) throws SQLException, Exception {
    ResultSet rs = null;
    try {
        String queryString = ("select * from `tema_3`.`" + tabel + "`");
        Statement stmt = con.createStatement(/*ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY*/);
        rs = stmt.executeQuery(queryString);
    } catch (SQLException sqle) {
        error = "SQLException: Interogarea nu a fost posibila.";
        throw new SQLException(error);
    } catch (Exception e) {
        error = "A aparut o exceptie in timp ce se extrageau datele.";
        throw new Exception(error);
    }
    return rs;
} // vedeTabela()
```

Se creeaza instructiunea de interogare SQL: "select \* from `tema\_3`"

Se realizeaza un statement pe conexiunea la baza de date, apoi se executa interogarea efectiva. Si astfel putem pune in evidenta partea de "back-end" din spatele comenzii din tabela de autori:

```
28      jb.connect();
29      ResultSet rs = jb.vedeTabela("autori");
30      long x;
31      while (rs.next()) {
32          x = rs.getInt("idautori");
33      }
34
35      <tr>
36          <td><input type="checkbox" name="primarykey" value="<%= x%>" /></td><td><%= x%></td>
37          <td><%= rs.getString("nume")%></td>
38          <td><%= rs.getString("prenume")%></td>
39          <td><%= rs.getString("adresa")%></td>
40          <td><%= rs.getString("varsta")%></td>
41      </tr>
42      }
43      %>
```

Incepand cu linia 31 cu structura repetitiva while, se plaseaza "cursorul" pe prima linie, si se preia din "idautori" valoarea id-ului ce se leaga de valorile pe care dorim sa le afisam.

Vom obtine prin apasarea butonului Vizualizare tabela Autori din cadrul index-ului urmatoarea pagina:

**Tabela Autori:**

[Adauga un nou autor.](#) [Home](#)

Mark:	IdAutor:	Nume:	Prenume:	Adresa:	Varsta:
<input type="checkbox"/>	1	Ionut	Ciprian	Ploiesti	35
<input type="checkbox"/>	2	Andrei	George	Bucuresti	45
<input type="checkbox"/>	3	Ana	Maria	Constanta	36
<input type="checkbox"/>		Popescu	Marin	Pitesti	28
<input type="checkbox"/>		Cândea	Sebastian Constantin	LEU A	17

[Home](#)

In spatele checkbox-urilor vom avea asociate valorile "idautori" specific fiecarui autor inregistrat. S-a realizat afisarea linie cu linie.

Cum se realizeaza stergerea?

Luam drept exemplu stergerea unui proiect. Ne uitam catre fisierul sterge\_proiect.jsp.

## Tabela Proiecte:

[Adauga un nou proiect.](#) [Home](#)

Mark:	IdProiect:	Denumire:	Tematica:	Termen limita:
<input type="checkbox"/>	1	Reolutia	Istorie	2024-12-02
<input type="checkbox"/>	2	Comunism	Istorie	2023-12-05
<input type="checkbox"/>	3	Colorat	Colorat	2025-08-01
<input type="checkbox"/>	4	Python	Programare	2023-06-23
<input checked="" type="checkbox"/>	7	Scoala	Educatie	2025-12-12

Sterge liniile marcate

[Home](#)

Daca alegem sa apasam butonul “Sterge liniile marcate” din cadrul paginii ce contine datele proiectelor, se va face legatura cu fisierul sterge\_proiect.jsp.

```
167 <form action="sterge_proiect.jsp" method="post">
```

Prin metoda POST, valorile ascunse in spatele checkbox-urilor sunt transmise prin functia `getParameter()`.

Se memoreaza toate datele pe care le-am bifat si se plaseaza in vectorul `s`, si prin obiectul `jb` se apeleaza la functia `stergeDateTabela(...)` din clasa `JavaBean`, functie ce are ca attribute `tabela` si totodata campul `dupa` care se doreste sa se faca stergerea.

```
167 public void stergeDateTabela(String[] primaryKeys, String tabela, String dupaID) throws SQLException, Exception {
168     if (con != null) {
169         try {
170             // creeaza un "prepared SQL statement"
171             long aux;
172             PreparedStatement delete;
173             delete = con.prepareStatement("DELETE FROM " + tabela + " WHERE " + dupaID + "=?");
174             for (int i = 0; i < primaryKeys.length; i++) {
175                 aux = java.lang.Long.parseLong(primaryKeys[i]);
176                 delete.setLong(1, aux);
177                 delete.execute();
178             }
179         } catch (SQLException sqle) {
180             error = "ExceptieSQL: Reactualizare nereusita; este posibil sa existe duplicate.";
181             throw new SQLException(error);
182         } catch (Exception e) {
183             error = "A aparut o exceptie in timp ce erau sterse inregistrarile.";
184             throw new Exception(error);
185         }
186     } else {
187         error = "Exceptie: Conexiunea cu baza de date a fost pierduta.";
188         throw new Exception(error);
189     }
190 } // end of stergeDateTabela()
191
```

Prin instructiunea if (con != null) se verifica daca avem conexiune cu baza de date. Apoi se declara un obiect "delete" de tip PreparedStatement, declaratie ce se realizeaza pe conexiunea con. Se insereaza instructiunea specifica MySQL urmata de "=?;" ce pune in evidenta faptul ca noi nu cunoastem cate id-uri au fost selectate. Prima valoare ce se ascunde sub "?" se inlocuieste cu valoarea aux, dupa care se executa si apoi se sterge din baza de date intreaga linie. Acest ciclu se continua pana cand se sterg toate datele ce au fost selectate. La finalul operatiei, se va afisa urmatoarea pagina:

### Proiect sters din baza de date

[Home](#)



Cum se adauga date?

Ca exemplu vom lua tabela asociere, cu fisierul JSP nou\_asociere.jsp

```
<%  
    int idautori, idproiecte;  
    String id1, id2, numeautor, prenumeautor, varsta, denumireproiect, tematicaproiect;  
    //aci  
    id1 = request.getParameter("idautori_asociere");  
    id2 = request.getParameter("idproiecte_asociere");  
    if (id1 != null && id2 != null ) {  
        jb.connect();  
        jb.adaugaAsociere(java.lang.Integer.parseInt(id1), java.lang.Integer.parseInt(id2));  
        jb.disconnect();  
    }  
%>
```

Se declara initial elementele pentru campurile pe care dorim sa le completam. Se preiau parametrii ce actioneaza ca si foreign-keys cu functiile de getParameter(...) si anume: idautori si idproiecte. Acestea sunt foarte importante deoarece parcurg autorii si proiectele pe care le doresc asociate datelor unei asocieri: SELECT.

Pagina in cadrul careia se adauga datele este:

## Suntem in tabela asocieri.

IdAutor: Selectati autorul:  ▼

IdProiect: Selectati proiectul:  ▼

[Home](#)

Prin apasarea butonul “Creeaza asociere”, se va face legatura cu functia `adaugaAsociere(...)` din clasa JavaBean:

```
public void adaugaAsociere(int idautori, int idproiecte)
    throws SQLException, Exception {
    if (con != null) {
        try {
            // creeaza un "prepared SQL statement"
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate("insert into asociere(idautori, idproiecte) values('" + idautori + "' , '" + idproiecte + "')");
            // se poate modifica valoarea datei astfel incat sa se ia data curenta a sistemului:
            // stmt.executeUpdate("insert into consultatie(idpacient, idmedic, DataConsultatie, Diagnostic, Medicament) values('" + idautori + "' , '" + idproiecte + "')");
        } catch (SQLException sqle) {
            error = "ExceptieSQL: Reactualizare nereusita; este posibil sa existe duplicate.";
            throw new SQLException(error);
        }
    } else {
        error = "Exceptie: Conexiunea cu baza de date a fost pierduta.";
        throw new Exception(error);
    }
} // end of adaugaAsociere
```

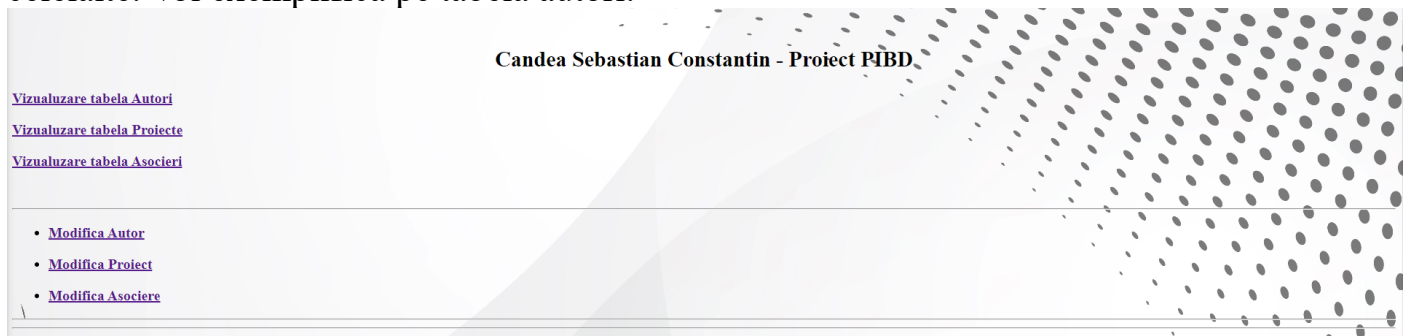
Asemenea operatiilor anterioare, se verifica daca avem conexiune cu baza de date si se executa o comanda specifica MySQL ( `insert into asociere(...)`).





Cum se realizeaza modificarea datelor?

Pe interfata site-ului, operatia de modificare a tabelelor are butoane separate fata de celelalte. Voi exemplifica pe tabela autori.



Apasand pe butonul Modifica Autor vom fi redirectionati catre fisierul modificare\_autor.jsp.

```
<%
    jb.connect();
    ResultSet rs = jb.vedeTabela("autori");
    long x;
    while (rs.next()) {
        x = rs.getLong("idautori");
    }
%>
```

Pe aceasta pagina se va afisa un tabel cu toti autorii si datele lor asemenea tabela\_autori.jsp.



Dupa selectarea id-ului si apasarea butonului „Modifica linie” se va deschide fisierul m1\_autor.jsp.

```
public ResultSet intoarceLinieDupaId(String tabela, String denumireId, int ID) throws SQLException, Exception {
    ResultSet rs = null;
    try {
        // Executa interogarea
        String queryString = ("SELECT * FROM " + tabela + " where " + denumireId + "=" + ID + ";");
        Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(queryString); //sql exception
    } catch (SQLException sqle) {
        error = "SQLException: Interogarea nu a fost posibila.";
        throw new SQLException(error);
    } catch (Exception e) {
        error = "A aparut o exceptie in timp ce se extrageau datele.";
        throw new Exception(error);
    }
    return rs;
} // end of intoarceLinieDupaId()
```

Se apeleaza functia intoarceLinieDupaId() din clasa JavaBean. Aceasta selecteaza datele dupa id-ul autorului pe care dorim sa-l modificam.

```
<form action="m2_autor.jsp" method="post">
  <table align="center">
    <tr>
      <td align="right">IdAutor:</td>
      <td><input type="text" name="idautori" size="30" value="<%= aux%>" readonly/></td>
    </tr>
    <tr>
      <td align="right">Nume:</td>
      <td><input type="text" name="nume" size="30" value="<%= Nume%>"/></td>
    </tr>
    <tr>
      <td align="right">Prenume:</td>
      <td><input type="text" name="prenume" size="30" value="<%= Prenume%>"/></td>
    </tr>
    <tr>
      <td align="right">Adresa:</td>
      <td><input type="text" name="adresa" size="30" value="<%= Adresa%>"/></td>
    </tr>
    <tr>
      <td align="right">Varsta:</td>
      <td><input type="text" name="varsta" size="30" value="<%= Varsta%>"/></td>
    </tr>
  </table><p align="center">
    <input type="submit" value="Modifica linia">
  </p>
</form>
```

Se creaza campuri de INPUT pentru modificarea datelor autorilor ce este legat de fisierul m2\_autor.jsp, unde se realizeaza modificarea propriu-zisa.

```

<p align="center"><a href="nou_autori.jsp"><b>Adauga un nou autor.</b></a> <a href="in
<%
    jb.connect();
    int aux = java.lang.Integer.parseInt(request.getParameter("idautori"));
    String Nume = request.getParameter("nume");
    String Prenume = request.getParameter("prenume");
    String Adresa = request.getParameter("adresa");
    String Varsta = request.getParameter("adresa");

    String[] valori = {Nume, Prenume, Adresa, Varsta};
    String[] campuri = {"nume", "prenume", "adresa"};
    jb.modificaTabela("autori", "idautori", aux, campuri, valori);
    jb.disconnect();
%>
<h1 align="center">Modificarile au fost efectuate !</h1>
<p align="center">
    <a href="index.html"><b>Home</b></a>

```

## 6) Concluzii

Bazele de date sunt foarte folosite la nivel global de fiecare companie, afacere care se ocupa cu productia are nevoie de baze de date pentru a-si implementa sistemul de lucru. Pentru a face o baza de date cat mai usor de modificat in viitor este important sa avem in vedere realizarea unei arhitecturi ce ofera posibilitatea de a separa functionalitatile.

Baza de date creata si interfata creata cu tehnologia JSP si prezentata in proiectul de mai sus permite unui utilizator sa efectueze operatiile cerute pe baza de date creata in MySQL: afisare, adaugare, modificare si stergere.

## 7) Bibliografie

- <https://www.nav.ro/blog/ce-este-mysql/> [1]
- <https://ro.wikipedia.org/> [2]
- Cursuri PIBD [3]
- <https://web.ceiti.md/lesson.php?id=1> [4]
- <https://ro.sawakinome.com/> [5]