Universitatea Politehnica din Bucuresti

Facultatea de Electronica,Telecomunicatii si Tehnologia Informatiei

Programarea Interfetelor Pentru Baze De Date

Tehnologie JSP

Elisei Stefan-Sergiu

Grupa-434C

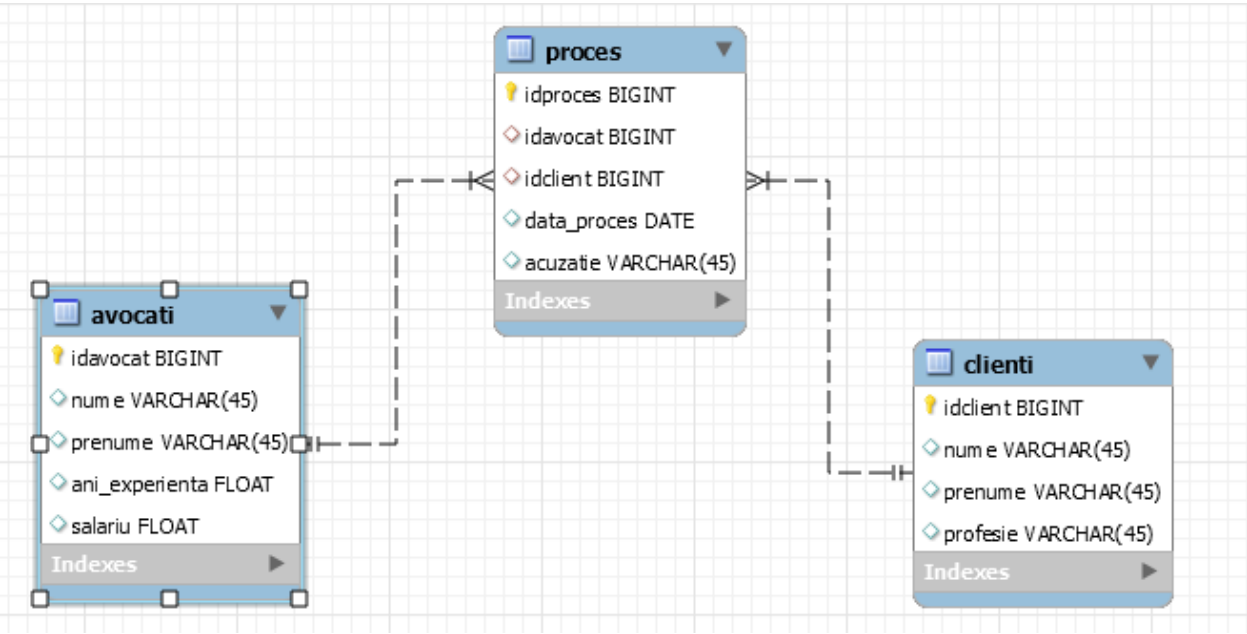
Cuprins

Baza de date

Tehnologia utilizata pentru realizarea bazei de date este MySQL.

MySQL este un sistem de gestionare a bazelor de date relaționale 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).

Datele găzduite în structură sunt capabile să recunoască relațiile dintre informațiile stocate. Fiecare bază de date conține tabele. Fiecare tabel (denumit și o relație) conține una sau mai multe categorii de date stocate în coloane (denumite și atribute). Fiecare rând (denumit, de asemenea, o înregistrare sau „tuple”) conține o informație unică (altfel menționată ca și cheie) pentru categoriile definite în coloane.[1]



Baza de date pe care am realizat-o contine 3 tabele:

- avocati
- clienti
- proces

Asocierea intre tabelele avocati si clienti este de tipul M:N. Astfel,s-a introdus o noua tabela (“proces”) ,cu ajutorul careia s-a simplificat asocierea la avocati M:1 proces 1:N clienti.

Atributele celor 3 tabele sunt:

1.avocati

- idavocat,tip BIGINT,primary key
- nume, tip VARCHAR(45)
- prenume, tip VARCHAR(45)
- ani_experienta,tip FLOAT
- salariu,tip FLOAT

2.clienti

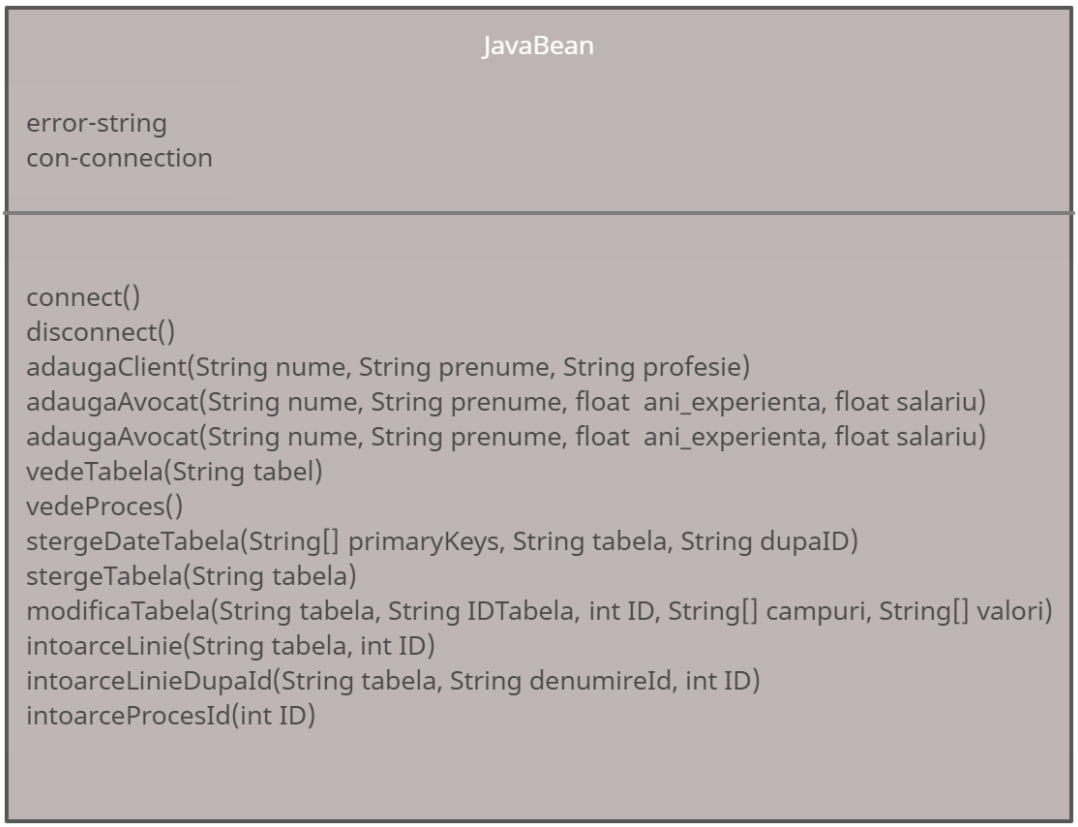
- idclient,tip BIGINT,primary key
- nume, tip VARCHAR(45)
- prenume, tip VARCHAR(45)
- profesie, tip VARCHAR(45)

3.proces

- idproces,tip BIGINT,primary key
- idavocat,tip BIGINT,foreign key
- idclient, tip BIGINT,foreign key
- data_proces,tip DATE
- acuzatie,tip VARCHAR(45)

Clasa JavaBean

Diagrama UML



JavaBean reprezinta clasa din care se preiau majoritatea functiilor si procedurilor pe care le foloseste interfata.

Astfel, conexiunea dintre interfata si baza de date se realizeaza cu ajutorul procedurei **connect()** cu care se incarca driverul MySql.Prin instanta con la Connection se conecteaza interfata la baza de date.

Daca aceasta conexiune nu este posibila, este returnata o eroare.

Asemanator, procedira **disconnect()** realizeaza deconectarea interfetei de la baza de date.

```
public void connect() throws ClassNotFoundException, SQLException, Exception {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/tema_pibd?useSSL=false", "root", "parolamysql");
    } 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()


public void disconnect() throws SQLException {
    try {
        if (con != null) {
            con.close();
        }
    } catch (SQLException sqle) {
        error = ("SQLException: Nu se poate inchide conexiunea la baza de date.");
        throw new SQLException(error);
    }
} // disconnect()
```

In clasa JavaBean se gasesc functii care realizeaza principalele operatii care se pot realiza intr-o baza de date:

- Vizualizare Tabela
- Adaugare date in tabela
- Modificare date in tabela
- Stergere date din tabela

Vizualizare Tabela

Aceasta operatie se realizeaza in clasa JavaBean cu ajutorul functiei vedeTabela, functie care foloseste ca parametru un sir “tabel” care va reprezenta numele tabelei pe care dorim sa o afisam.

Aceasta functie returneaza variabila rs de tip ResultSet care contine toate instantele din “tabel”.

Aceasta functie este echivalenta interogarii : “select * from “baza_de_date” “nume_tabela”;

Pentru Tabela proces, s-a realizat o functie separata “vedeProces”, in care se asigura faptul ca idavocat din tabela process este acelasi cu idavocat din tabela avocati si idclient din tabela process este acelasi cu idclient din tabela clienti, astfel realizandu-se un inner join.

```
public ResultSet vedeTabela(String tabel) throws SQLException, Exception {
    ResultSet rs = null;
    try {
        String queryString = ("select * from `tema_pibd`.`" + 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()
```

Adaugare date in tabela

Pentru toate cele 3 tabele s-a realizat cate o functie de adaugare in tabela.

Cu ajutorul instantei stmt la Statement, se executa functia **stmt.executeUpdate()** prin care se adauga date in tabela.

```
public void adaugaClient(String nume, String prenume, String profesie)
        throws SQLException, Exception {
    if (con != null) {
        try {
            // create a prepared SQL statement
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate("insert into clienti(nume, prenume, profesie) values('" + nume + "'  , '" + prenume + "', '" + profesie + "');");

        } 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 adaugaClient()
```

Modificare date in tabela

```
public void modificaTabela(String tabela, String IDTabela, int ID, String[] campuri, String[] valori) throws SQLException, Exceptior
    String update = "update " + tabela + " set ";
    String temp = "";
    if (con != null) {
        try {
            for (int i = 0; i < campuri.length; i++) {
                if (i != (campuri.length - 1)) {
                    temp = temp + campuri[i] + "='" + valori[i] + "', ";
                } else {
                    temp = temp + campuri[i] + "='" + valori[i] + "' where " + IDTabela + " = '" + ID + "';";
                }
            }
            update = update + temp;
            // create a prepared SQL statement
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate(update);
        } 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 modificaTabela()
```

Stergere date din tabela

In aceasta functie stergerea se realizeaza cu ajutorul instantei delete, Prin executarea functiei **delete.execute()**,functie care i se aplica lui aux, o variabila de tip long, care este id-ul corespunzator liniei pe care dorim sa o stergem.

```
public void stergeDateTabela(String[] primaryKeys, String tabela, String dupaID) throws SQLException, Exception {
    if (con != null) {
        try {
            // create a prepared SQL statement
            long aux;
            PreparedStatement delete;
            delete = con.prepareStatement("DELETE FROM " + tabela + " WHERE " + dupaID + "=?;");
            for (int i = 0; i < primaryKeys.length; i++) {
                aux = java.lang.Long.parseLong(primaryKeys[i]);
                delete.setLong(1, aux);
                delete.execute();
            }
        } catch (SQLException sqle) {
            error = "ExceptieSQL: Reactualizare nereusita; este posibil sa existe duplicate.";
            throw new SQLException(error);
        } catch (Exception e) {
            error = "A aparut o exceptie in timp ce erau sterse inregistrarile.";
            throw new Exception(error);
        }
    } else {
        error = "Exceptie: Conexiunea cu baza de date a fost pierduta.";
        throw new Exception(error);
    }
} // end of stergeDateTabela()
```

Intoarce Linie dupa ID

Aceasta functie va returna toate atributele unei linii pe care dorim sa o modificam. Aceasta linie este corespunzatoare ID-ului bifat.

```
public ResultSet intoarceLinieDupaId(String tabela, String denumireId, int ID) throws SQLException, Exception {
    ResultSet rs = null;
    try {
        // Execute query
        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()
```

Tehnologia utilizata:JSP

JSP-Java server pages

JAVA SERVER PAGES este este o simplă dar puternică tehnologie folosită pe partea de server pentru a genera continut HTML dinamic.

JSP este o extensie directă a Java Servlets și furnizează o modalitate de a separa partea de procesare de cea de prezentare. JSP-urile au acces la intreaga familie Java API (Application Programming Interface), incluzand si JDBC (Java Database Connectivity), cea din urma definind modul in care un utilizator poate accease baza de date.

O componenta JSP este creata pentru a prelua rolul de interfata cu utilizatorul pentru o aplicatie Java Web.

Dezvoltatorii Web combina in fisierele JSP cod HTML, elemente de tip XML, precum si actiuni si comenzi JSP incorporate.

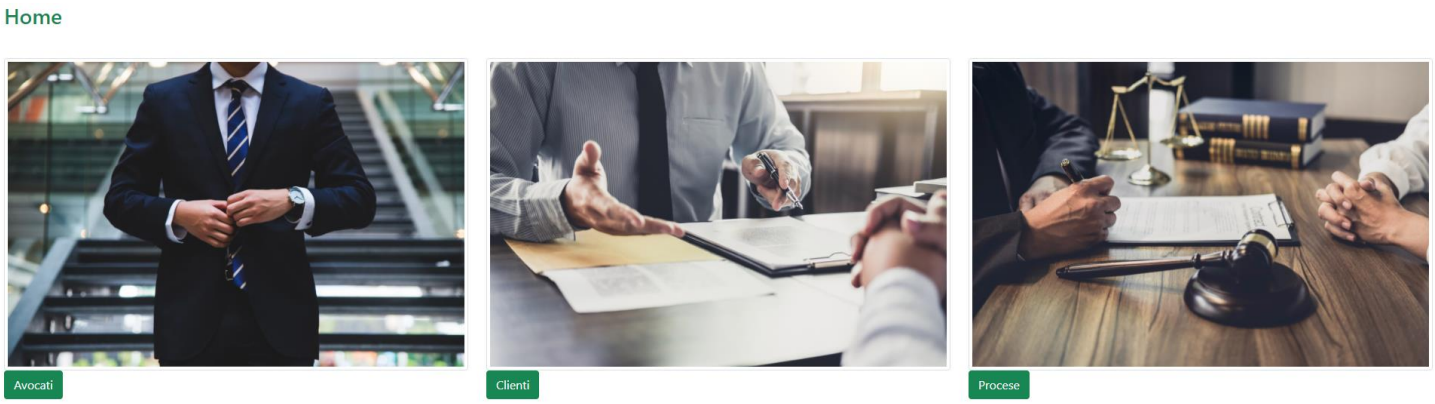
Folosind JSP se pot colecta date introduce de utilizatori prin formulare Web, se pot prezenta date dintr-o baza de date (sau dintr-o alta sursa), si se pot crea pagini Web dinamice. [2]

Interfata Web

Pagina principala a aplicatiei este implementata cu ajutorul fisierului index.html.

Designul l-am realizat cu ajutorul site-ului Layoutit.com[3]

Este accesat urmatorul url: http://localhost:8080/tema_pibd/index.html




```
<div class="row">

    <div class="col-md-4">
        <img alt="Bootstrap Image Preview" src="https://englishlawyersjapan.com/wp-content/uploads/2019/09/corporate-law.jpg" class="img-thumbnail">
        <a href="tabela_Avocati.jsp"><button type="button" class="btn btn-success">
            Avocati
        </button></a>
    </div>

    <div class="col-md-4">
        <img alt="Bootstrap Image Preview" src="https://scanners.us.fujitsu.com/-/media/project/scanners/blog/2019/08/istock-941906652.jpg?mw=1440&w=1440&hash=A01A16600BBF82E77F2
        <a href="tabela_Clienti.jsp"><button type="button" class="btn btn-success">
            Clienti
        </button></a>
    </div>
    <div class="col-md-4">
        <img alt="Bootstrap Image Preview" src="https://www.criminaldefencelawyers.com.au/wp-content/uploads/2020/02/Criminal-lawyer-in-consultation-with-client-1440x960.jpg" cla
        <a href="tabela_Proces.jsp"><button type="button" class="btn btn-success">
            Procese

        </button></a>
    </div>
</div>
<p></p>

<div class="row">
    <div class="col-md-12">
         <p>
        <a href="modificari.jsp"><button type="button" class="btn btn-success" style="margin-top: 100px">
            Modificari
        </button>
        </a>
        </p>
        <p>
        <a href="stergeri.jsp"><button type="button" class="btn btn-success">
            Stergeri
        </button></a>

        </p>
```

Apasand pe unul dintre butoanele corespunzatoare tabelelor din baza de date, se vor vizualiza pe ecran datele existente in tabele.De exemplu,la apasarea butonului “Avocati”, se apeleaza fisierul tabela_Avocati.jsp, in interiorul caruia se realizeaza conectarea cu baza de date si apelarea functiei vedeTabela(avocati)

```
68              </thead>
69              <%
70                  jb.connect();
71                  ResultSet rs = jb.vedeTabela("avocati");
72                  long x;
73                  while (rs.next()) {
74                      x = rs.getInt("idavocat");
75              %>
76              <tr>
77                 <td><%= x%></td>
78                  <td><%= rs.getString("Nume")%></td>
79                  <td><%= rs.getString("Prenume")%></td>
80                  <td><%= rs.getFloat("ani_experienta")%></td>
81                  <td><%= rs.getFloat("salariu")%></td>
82                  <%
83                      }
84                  %>
85              </tr>
```

Tabela Avocati

Home

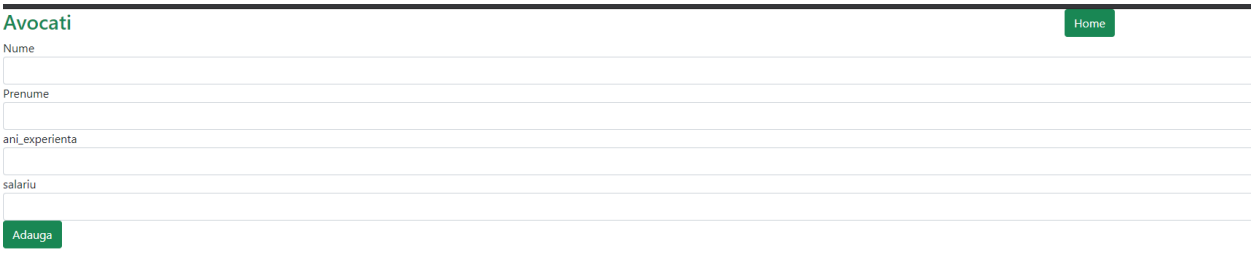
Adauga avocat nou

idavocat	Nume	Prenume	ani_experienta	salariu
39	Elisei	Stefan	10.0	9000.0
40	Popescul	Valentin	7.0	1999.0
41	Busila	Leonora	6.0	2800.0

Pe ecran apar afisate toate atributele tabelei, precum si operatia de adaugare de date noi si de intoarcere la pagina principala.

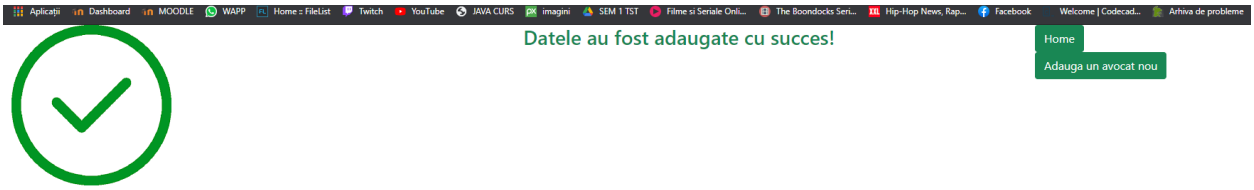
Operatia de adaugare de date noi se realizeaza prin fisierul nou_Avocat, in care se cere introducerea variabilelor nume,prenume,ani_experienta,salariu,variabile ce vor fi folosite in functia adaugaAvocat.

```
<body>
    <%

        String Nume = request.getParameter("Nume");
        String Prenume = request.getParameter("Prenume");
        String ani_experienta2 = request.getParameter("ani_experienta");
        String salariu2 = request.getParameter("salariu");


        if (Nume != null) {
            Float ani_experienta=Float.parseFloat(ani_experienta2);
            Float salariu=Float.parseFloat(salariu2);
            jb.connect();
            jb.adaugaAvocat(Nume, Prenume, ani_experienta,salariu);
            jb.disconnect();
    %>
```

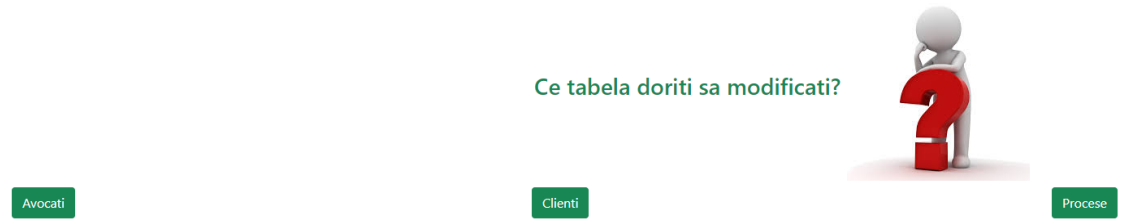


La adaugarea de noi date in tabela, apar campuri care trebuie completate cu datele noi pe care dorim sa le introducem.

Odata introduse datele, se apasa butonul de adauga si se primeste un mesaj de confirmare a datelor introduse.(



Pentru operatiile de **modificare** si **stergere**, am creat 2 butoane, pe care odata la apasarea unuia dintre ele, utilizatorul este interogat pe care tabela doreste realizarea operatiei.



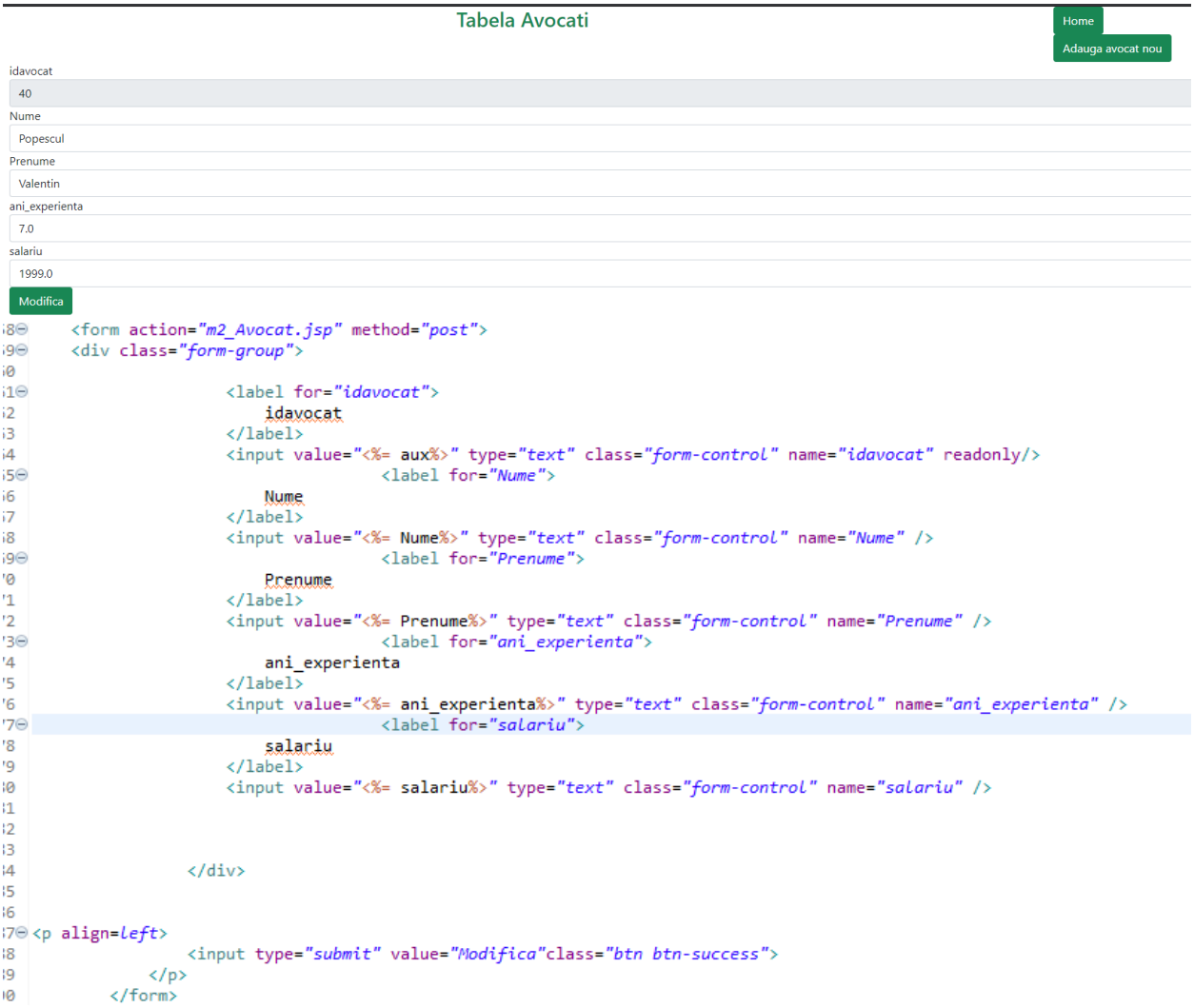
Pentru modificare exista 3 fisiere jsp:

- **modifica_”tabela”.jsp**(unde tabela este numele tabelei)
  Aici se afiseaza tabela cu un checkbox, pentru alegerea liniei pe care dorim sa o modificam

```
    <div class="row">
        <div class="col-md-12">
        <form action="m1_Avocat.jsp" method="post">
            <table class="table table-sm table-hover table-striped">
                <thead>
                    <tr>
                        <th>
                            Mark
                        </th>
                        <th>
                            idavocat
                        </th>
                        <th>
                            Nume
                        </th>
                        <th>
                            Prenume
                        </th>
                        <th>
                            ani_experienta
                        </th>
                        <th>
                            salariu
                        </th>
                    </tr>
                </thead>
                <%
                    jb.connect();
                    ResultSet rs = jb.vedeTabela("avocati");
                    long x;
                    while (rs.next()) {
                        x = rs.getInt("idavocat");
                %>
                <tr>
                    <td><input type="checkbox" name="primarykey" value="<%= x%>" /></td><td><%= x%></td>
                    <td><%= rs.getString("Nume")%></td>
                    <td><%= rs.getString("Prenume")%></td>
                    <td><%= rs.getFloat("ani_experienta")%></td>
                    <td><%= rs.getFloat("salariu")%></td>
                    <%
                        }
                    %>
                </tr>
            </table>
```

- **m1_”tabela”.jsp**

  Aici se afiseaza datele din linia pe care dorim sa o modificam, date pe care le putem modifica direct pe campuri.



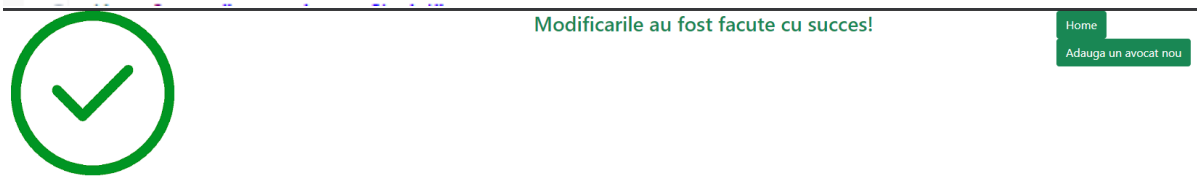

```
    <form action="m2_Avocat.jsp" method="post">
    <div class="form-group">

                    <label for="idavocat">
                        idavocat
                    </label>
                    <input value="<%= aux%>" type="text" class="form-control" name="idavocat" readonly/>
                                <label for="Nume">
                        Nume
                    </label>
                    <input value="<%= Nume%>" type="text" class="form-control" name="Nume" />
                                <label for="Prenume">
                        Prenume
                    </label>
                    <input value="<%= Prenume%>" type="text" class="form-control" name="Prenume" />
                                <label for="ani_experienta">
                        ani_experienta
                    </label>
                    <input value="<%= ani_experienta%>" type="text" class="form-control" name="ani_experienta" />
                                <label for="salariu">
                        salariu
                    </label>
                    <input value="<%= salariu%>" type="text" class="form-control" name="salariu" />



                </div>


<p align=left>
                <input type="submit" value="Modifica"class="btn btn-success">
            </p>
        </form>
```

  La apasarea butonului modifica se apeleaza m2_tabela.jsp

- **m2_”tabela”.jsp**

  Aici se realizeaza operatia de modificare a liniei, se apeleaza functia *modificaTabela()*, si se afiseaza mesajul de confirmare a modificarilor.

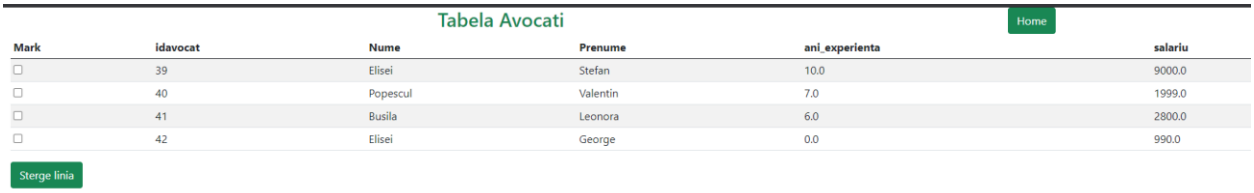
```jsp
<body>
        <%
        jb.connect();

        int aux = java.lang.Integer.parseInt(request.getParameter("idavocat"));
        String Nume = request.getParameter("Nume");
        String Prenume = request.getParameter("Prenume");
        String ani_experienta = request.getParameter("ani_experienta");
        String salariu = request.getParameter("salariu");


        String[] valori = {Nume, Prenume, ani_experienta, salariu};


        String[] campuri = {"Nume", "Prenume","ani_experienta","salariu"};
        jb.modificaTabela("avocati", "idavocat", aux, campuri, valori);
        jb.disconnect();
    %>
```



Pentru steregerea datelor, exista 2 fisiere jsp:

- tabela_stergere_”tabela”.jsp

  Aici se afiseaza tabela in care dorim sa modificam o linie impreuna cu un checkbox, corespunzator liniei pe care dorim sa o modificam

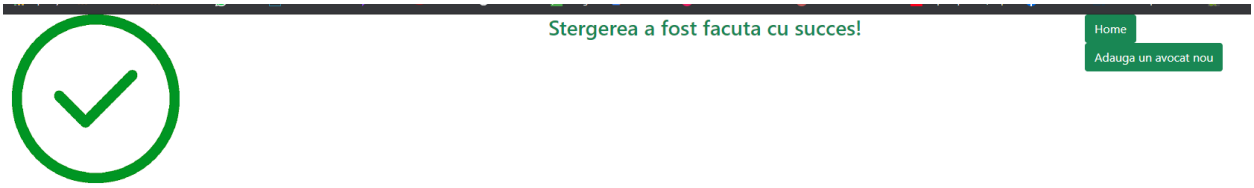


- sterge_”tabela”.jsp

  In acest fisier jsp se realizeaza stergerea propriu-zisa si afisarea mesajului de confirmare pentru stergere.

  Se apeleaza functia *stergeDateTabela()* pentru linia corespunzatoare checkboxului bifat in fisierul anterior.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
        <title>Tabela Avocati</title>
    </head>
    <jsp:useBean id="jb" scope="session" class="db.JavaBean" />
    <jsp:setProperty name="jb" property="*" />
    <body>
        <%
            String[] s = request.getParameterValues("primarykey");
            jb.connect();
            jb.stergeDateTabela(s, "avocati", "idavocat");
            jb.disconnect();
        %>
```



Bibliografie

- https://www.nav.ro/blog/ce-este-mysql/
- https://ro.scribd.com/doc/147179302/Tehnologia-JSP
- https://www.layoutit.com/