Proiect realizat pentru disciplina
PROGRAMAREA INTERFEȚELOR PENTRU BAZE DE DATE

Java Server Pages

Student
Tucă Mihnea-Luca-Ioan
434B

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

Universitatea Națională de Știință și Tehnologie Politehnică București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
February, 2024

Contents

List of Figures

1 Introducere

1.1 Tema proiectului

Tema proiectului se bazează pe dezvoltarea unei aplicații ce conține o bază de date, creată în sistemul de gestionare a bazelor de date MySQL si își propune să demonstreze flexibilitatea și eficiența tehnologiei JSP[6] prin implementarea operațiilor CRUD - Create, Read, Update, Delete - într-o aplicație web interactivă.

1.2 Obiective

Acest proiect își propune următoarele obiective:

Gestionarea Muzicienilor

- Crearea de noi înregistrări pentru muzicieni.
- Vizualizarea detaliilor muzicienilor existenți.
- Actualizarea informațiilor muzicienilor.
- Ștergerea înregistrărilor muzicienilor inactivi sau duplicați.

Administrarea Stilurilor Muzicale

- Adăugarea de noi stiluri muzicale în sistem.
- Afișarea listei stilurilor muzicale, cu detalii precum origini și caracteristici.
- Modificarea informațiilor despre stilurile muzicale (de exemplu, actualizarea descrierii).
- Eliminarea stilurilor muzicale care nu mai sunt relevante.

Organizarea Concertelor

- Planificarea concertelor pentru diferiți muzicieni sau grupuri.
- Vizualizarea detaliată a concertelor, inclusiv locația, data și ora.
- Actualizarea detaliilor concertelor, cum ar fi schimbarea locației sau a datei.
- Anularea concertelor programate.

1.3 Aplicabilitate

Aplicația poate fi folosită pentru a gestiona carierele muzicienilor și evenimentele live. Tabela "Muzicieni" stochează informații despre artiști, în timp ce "Stiluri Muzicale" și "Concerte" sunt utilizate pentru a clasifica genurile muzicale și pentru a organiza evenimentele. Utilizatorii pot gestiona cu ușurință programările, promovarea stilurilor muzicale și coordonarea detaliilor concertelor, oferind o platformă centralizată pentru dinamica industriei muzicale.

2 Tehnologii utilizate

2.1 Sistemul de gestiune a bazelor de date MySQL

MySQL[2] este un sistem de gestiune a bazelor de date relaționale (SGBDR), open-source, care folosește Structured Query Language (SQL) pentru gestionarea datelor. Este ales pentru flexibilitatea, fiabilitatea și performanța sa, fiind ideal atât pentru aplicații mici, cât și pentru sisteme mari și complexe. MySQL excellează în aplicații web datorită integrării sale ușoare cu limbaje de programare precum Java, oferind o soluție robustă pentru dezvoltarea web dinamică.

2.2 JSP

Java Server Pages (JSP)[6] este o tehnologie esențială pentru dezvoltarea aplicațiilor web, oferind un amestec perfect între HTML și codul Java. Prin capacitatea sa de a genera conținut web dinamic și posibilitatea de integrare cu JavaBeans și alte tehnologii Java, cum ar fi JDBC, JSP devine o alegere ideală pentru crearea de aplicații web complexe și interacțiuni bazate pe date. Avantajele sale, cum ar fi portabilitatea, scalabilitatea și eficiența în gestionarea interfețelor utilizator dinamice, îl fac nu doar practic, ci și o opțiune populară printre dezvoltatorii care doresc să creeze experiențe web bogate și personalizate.

2.3 HTML

HTML (HyperText Markup Language)[5] este limba de bază pentru crearea și structurarea paginilor web. Folosind o serie de elemente și atribute, HTML permite dezvoltatorilor să definească structura de conținut, cum ar fi paragrafe, titluri, linkuri și imagini, pe o pagină web. Această structură semantică nu numai că este esențială pentru prezentarea vizuală și organizarea conținutului în browser, dar joacă un rol crucial în accesibilitatea și optimizarea motorului de căutare (SEO). Fiind strâns integrată cu CSS pentru stilizare și JavaScript pentru funcționalități, HTML stă la baza dezvoltării web, permițând crearea de site-uri web interactive și atractive.

2.4 Apache Tomcat

Apache Tomcat este un container de servlet-uri open-source implementat în Java, care permite rularea aplicațiilor web dezvoltate cu tehnologia Java Servlet și JavaServer Pages (JSP). Este unul dintre cele mai populare servere web pentru aplicații Java, datorită ușurinței de utilizare, flexibilității și performanței solide. Tomcat acționează ca un server web sau un mediul de execuție pentru aplicațiile Java, gestionând cererile HTTP de la clienți și facilitând executarea servlet-urilor și JSP-urilor pentru generarea răspunsurilor dinamice

3 Descrierea aplicației

3.1 Creearea bazei de date

Tema se bazează pe crearea unei baze de date ce are doua tabele în asociere M:N, "Muzicieni" și "StiluriMuzicale".

Table Name: muzicieni Schema: mihnea

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idmuzician	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
nume	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
prenume	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
datanasterii	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
taraorigine	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
instrumentprincipal	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure 3.1: Tabela Muzicieni

Table Name: stiluri_muzicale Schema: mihnea

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idstil	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
numestil	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
originestil	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
perioadapopularitate	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure 3.2: Tabela Stiluri Muzicale

Asocierea M:N (Many-to-Many) este o relație de bază de date între două tabele care permite fiecărei înregistrări din prima tabelă să fie asociate cu mai multe înregistrări din a doua tabelă și invers. Această relație necesită crearea unei tabele intermediare pentru a gestiona această asociere. În cazul nostru, am creat tabela de legătura "Concerte". Am ales ca și cheie primara idconcert. În această nouă tabelă, atributele ce au fost selectate ca și chei primare pentru tabelele anterioare, vor deveni chei străine (FK) pentru tabela de legătură.[2]

Table Name: concerte Schema: mihnea

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idconcert	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
idmuzician	BIGINT	☐	☑	☐	☐	☐	☐	☐	☐	
idstil	BIGINT	☐	☑	☐	☐	☐	☐	☐	☐	
dataconcert	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
locatie	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
altedetalii	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure 3.3: Tabela Concerte

Diagrama EER (Enhanced Entity-Relationship) a unei baze de date este o reprezentare vizuală complexă care ilustrează entitățile, relațiile și constrângerile acestora, oferind o vedere detaliată și extinsă a structurii și interconexiunilor din cadrul bazei de date.

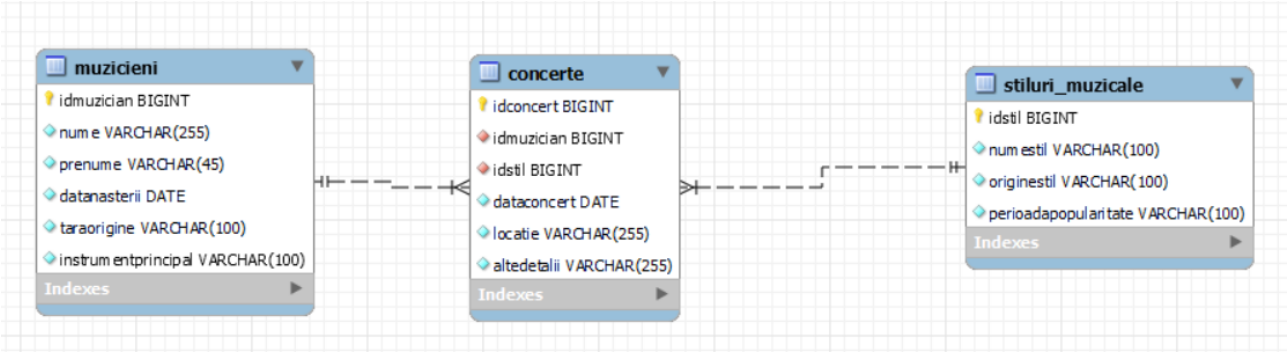

Figure 3.4: Diagrama EER (Enhanced Entity-Relationship)

3.2 Structura fisierelor

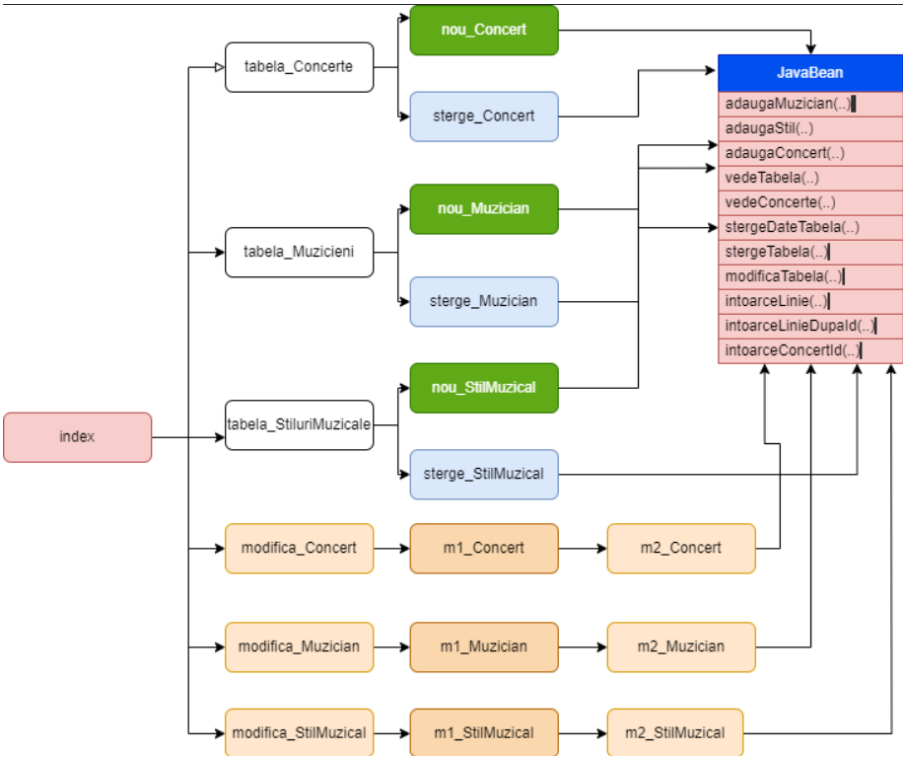

Figure 3.5: Diagrama implementării fișierelor ce alcătuiesc pagina web

3.3 Implementarea clasei JavaBean

Clasa JavaBean servește ca un strat de intermediere între aplicația Java și baza de date MySQL, oferind metode pentru conectarea la baza de date, manipularea datelor și gestionarea excepțiilor.[1]

3.3.1 Inițializare și Conectare la Baza de Date:

Metodele connect(), connect(String bd) și connect(String bd, String ip) sunt utilizate pentru a stabili conexiunea cu baza de date MySQL, folosind driverul JDBC și credențialele specificate.

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

Figure 3.6: Metoda connect

3.3.2 Închiderea Conexiunii:

Metoda disconnect() închide conexiunea cu baza de date, asigurând gestionarea corespunzătoare a resurselor.

```
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()
```

Figure 3.7: Metoda disconnect

3.3.3 Manipularea Datelor:

Metodele adaugaMuzician(), adaugaStil() și adaugaConcert() sunt folosite pentru a insera noi înregistrări în tabelele corespunzătoare.

```
public void adaugaMuzician(String nume, String prenume, String datanasterii, String taraorigine,
        String instrumentprincipal)
        throws SQLException, Exception {
    if (con != null) {
        try {
            // create a prepared SQL statement
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate("INSERT INTO muzicieni (nume, prenume, datanasterii, taraorigine, "
                    + "instrumentprincipal) VALUES ('" + nume + "', '" + prenume + "', '"
                    + datanasterii + "', '" + taraorigine + "', '" + instrumentprincipal + "');");

        } 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);
    }
}
```

Figure 3.8: Metoda adaugaMuzician

vedeTabela(String tabel) și vedeHotel() permit extragerea și vizualizarea datelor din tabele.

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

Figure 3.9: Metoda vedeTabela

Pentru tabela de legătură "Concerte", se realizează un CROSS JOIN între tabele.

```java
public ResultSet vedeConcerte() throws SQLException, Exception {
    ResultSet rs = null;
    try {
        String queryString = "SELECT c.idconcert, c.dataconcert, c.locatie,"
                + " c.altedetalii, m.idmuzician, m.nume, m.prenume, m.datanasterii,"
                + " m.taraorigine, m.instrumentprincipal, sm.idstil, sm.numestil, sm.originestil, "
                + "sm.perioadapopularitate FROM concerte c, muzicieni m, stiluri_muzicale sm"
                + " WHERE c.idmuzician = m.idmuzician AND c.idstil = sm.idstil;";
        Statement stmt = con.createStatement(/*ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_R
        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;
}
```

Figure 3.10: Metoda vedeConcert

stergeDateTabela() este utilizata pentru a șterge înregistrări.

```
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);
    }
}
```

Figure 3.11: Metoda stergeDateTabela

modificaTabela() permite actualizarea înregistrărilor dintr-o tabelă specificată. intoarceLinie() și intoarceLinieDupaId() sunt folosite pentru a extrage rânduri specifice din tabele.

```
public void modificaTabela(String tabela, String IDTabela, int ID, String[] campuri, String[] valori)
        throws SQLException, Exception {
    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()
```

Figure 3.12: Metoda modificaTabela

```java
public ResultSet intoarceLinie(String tabela, int ID) throws SQLException, Exception {
    ResultSet rs = null;
    try {
        // Execute query
        String queryString = ("SELECT * FROM " + tabela + " where idmuzician=" + 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;
}
```

Figure 3.13: Metoda intoarceLinie

```java
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()
```

Figure 3.14: Metoda intoarceLinieDupaId

3.3.4 Gestionarea Excepțiilor:

Fiecare metodă de conectare și manipulare a datelor este concepută pentru a gestiona și arunca excepții specifice, cum ar fi ClassNotFoundException, SQLException, și Exception. Aceasta asigură că erorile sunt tratate corespunzător și că aplicația poate răspunde adecvat la problemele de conectivitate sau la alte probleme care pot apărea.[1]

3.4 Implementarea JSP-urilor

3.4.1 Afișarea datelor

Urmatorul fragment de cod JSP este folosit pentru a afișa și interacționa cu datele referitoare la muzicieni, permițând utilizatorilor să selecteze și să execute acțiuni pe date specifice muzicianului.

```html
<table class="table table-bordered">
    <tr>
        <td><b>Mark:</b></td>
        <td><b>IdMuzician:</b></td>
        <td><b>Nume:</b></td>
        <td><b>Prenume:</b></td>
        <td><b>DataNasterii:</b></td>
        <td><b>TaraOrigine:</b></td>
        <td><b>InstrumentPrincipal:</b></td>
    </tr>
```

Figure 3.15: Fragment de cod html pentru afișarea datelor din tabela Muzicieni

```jsp
<%
    jb.connect();
    ResultSet rs = jb.vedeTabela("muzicieni");
    long x;
    while (rs.next()) {
        x = rs.getInt("idmuzician");
%>
<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.getDate("datanasterii")%></td>
    <td><%= rs.getString("taraorigine")%></td>
    <td><%= rs.getString("instrumentprincipal")%></td>
</tr>
<%
    }
%>
```

Figure 3.16: Fragment de cod java si html pentru afișarea datelor din tabela Muzicieni

Figure 3.17: Pagina JSP pentru afișare

3.4.2 Adăugarea datelor

Adăugarea se realizează prin apăsarea butonului **Adaugă** din pagina web. Acest fragment de cod JSP se ocupă cu preluarea parametrilor dintr-o cerere HTTP și utilizarea acestor parametri pentru a adăuga un nou muzician într-o bază de date.

```
<%
    String nume = request.getParameter("nume");
    String prenume = request.getParameter("prenume");
    String datanasterii = request.getParameter("datanasterii");
    String taraorigine = request.getParameter("taraorigine");
    String instrumentprincipal = request.getParameter("instrumentprincipal");
    if (nume != null && prenume != null && datanasterii != null && taraorigine != null &&
            instrumentprincipal != null) {
        jb.connect();
        jb.adaugaMuzician(nume, prenume, datanasterii, taraorigine, instrumentprincipal);
        jb.disconnect();
%>
<p class="alert alert-success">Muzicianul a fost adaugat.</p><%
    } else {
%>
```

Figure 3.18: Fragment de cod pentru preluarea datelor

```
<form action="nou_Muzician.jsp" method="post">
    <div class="form-group">
        <label for="nume">Nume:</label>
        <input type="text" class="form-control" id="nume" name="nume" />
    </div>
    <div class="form-group">
        <label for="prenume">Prenume:</label>
        <input type="text" class="form-control" id="prenume" name="prenume" />
    </div>
    <div class="form-group">
        <label for="datanasterii">Data Nasterii:</label>
        <input type="text" class="form-control" id="datanasterii" name="datanasterii" />
    </div>
    <div class="form-group">
        <label for="taraorigine">Tara de Origine:</label>
        <input type="text" class="form-control" id="taraorigine" name="taraorigine" />
    </div>
    <div class="form-group">
        <label for="instrumentprincipal">Instrument Principal:</label>
        <input type="text" class="form-control" id="instrumentprincipal" name="instrumentprincipal" /
    </div>
    <button type="submit" class="btn btn-primary">Adauga muzicianul</button>
</form>
```

Figure 3.19: Fragment de cod pentru preluarea datelor dintr-un formular

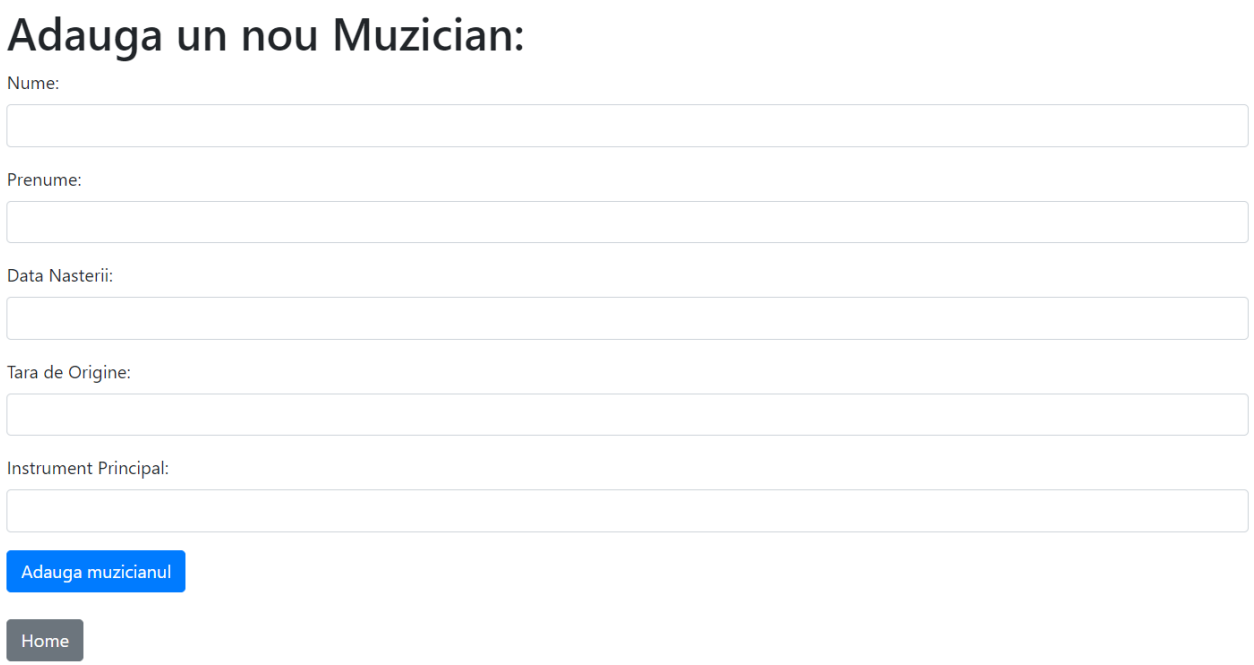

Figure 3.20: Formularul de preluare a datelor

3.4.3 Modificarea datelor

Modificarea datelor se realizează prin selectarea checkboxului și apăsarea butonului de **Modifica** din pagina. Acest fragment de cod JSP se ocupă cu preluarea parametrilor dintr-o cerere HTTP și utilizarea acestor parametri pentru a modifica muzicianul ales de utilizator din bază de date.

```
<%
    jb.connect();
    int aux = java.lang.Integer.parseInt(request.getParameter("idmuzician"));
    String nume = request.getParameter("nume");
    String prenume = request.getParameter("prenume");
    String datanasterii = request.getParameter("datanasterii");
    String taraorigine = request.getParameter("taraorigine");
    String instrumentprincipal = request.getParameter("instrumentprincipal");

    String[] valori = {nume, prenume, datanasterii, taraorigine, instrumentprincipal};
    String[] campuri = {"nume", "prenume", "datanasterii", "taraorigine", "instrumentprincipal"};
    jb.modificaTabela("muzicieni", "idmuzician", aux, campuri, valori);
    jb.disconnect();
%>
```

Figure 3.21: Fragment de cod pentru modificarea datelor din tabela

```
<div class="form-group">
    <label for="nume" class="col-sm-2 control-label">Nume:</label>
    <div class="col-sm-10">
        <input type="text" class="form-control" id="nume" name="nume" value="<%= nume%>"/>
    </div>
</div>
<div class="form-group">
    <label for="prenume" class="col-sm-2 control-label">Prenume:</label>
    <div class="col-sm-10">
        <input type="text" class="form-control" id="prenume" name="prenume" value="<%= prenume%>"/>
    </div>
</div>
```

Figure 3.22: Formularul pentru preluarea datelor

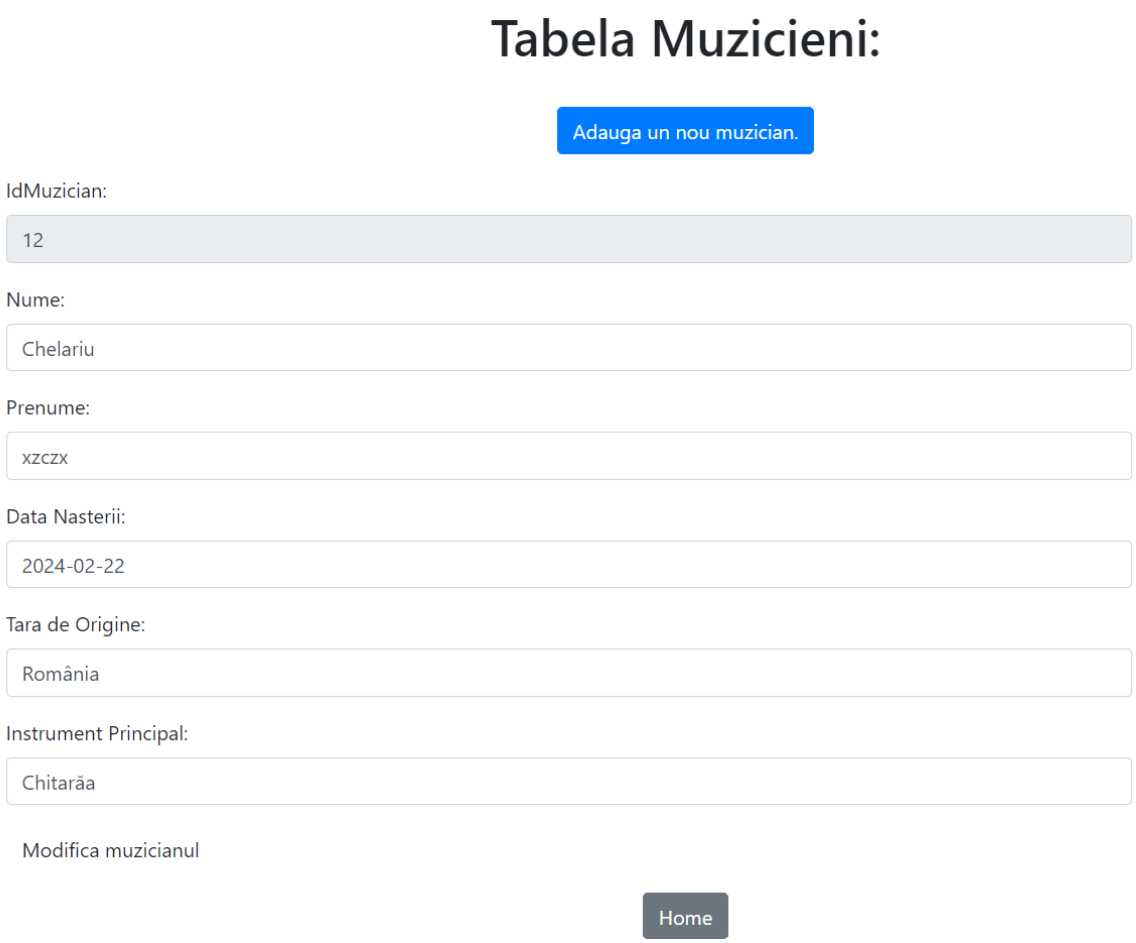

Figure 3.23: Formularul pentru modificarea datelor JSP

3.4.4 Ștergerea datelor

Ștergerea datelor se realizează prin selectarea checkboxului și apăsarea butonului de **Ștergere** din pagina.

```jsp
<%
    String[] s = request.getParameterValues("primarykey");
    jb.connect();
    jb.stergeDateTabela(s, "muzicieni", "idmuzician");
    jb.disconnect();
%>
```

Figure 3.24: Codul JSP pentru ștergerea datelor

3.5 Realizarea Frontend-ului folosind Bootstrap

Bootstrap[3] facilitează dezvoltarea web prin oferirea de stiluri și componente gata de folosit, care garantează un design receptiv și estetic. Este o alegere populară pentru a asigura coerența vizuală și funcționalitatea interactivă fără a scrie mult cod.

Cu plugin-uri JavaScript și o amplă documentație, Bootstrap ajută în crearea rapidă a site-urilor moderne, adaptabile la orice dimensiune de ecran, ideal pentru dezvoltatorii care doresc eficiență și accesibilitate.

```html
<body>
    <div class="container my-4">
        <hr>
        <h2 class="text-center my-4">Vizualizari + Adaugari + Stergeri</h2>
        <div class="row text-center">
            <div class="col-md-4">
                <a href="tabela_Concerte.jsp" class="btn btn-primary btn-block"><strong>Concerte</strong></a>
            </div>
            <div class="col-md-4">
                <a href="tabela_Muzicieni.jsp" class="btn btn-secondary btn-block"><strong>Muzicieni</strong></a>
            </div>
            <div class="col-md-4">
                <a href="tabela_StiluriMuzicale.jsp" class="btn btn-success btn-block"><strong>Stiluri Muzicale</strong></a>
            </div>
        </div>
        <hr>
        <ul class="list-unstyled">
            <li><a href="modifica_Concert.jsp" class="btn btn-info btn-block"><strong>Modifica Concerte</strong></a></li>
            <li><a href="modifica_Muzician.jsp" class="btn btn-info btn-block"><strong>Modifica Muzicieni</strong></a></li>
            <li><a href="modifica_StilMuzical.jsp" class="btn btn-info btn-block"><strong>Modifica Stiluri Muzicale</strong></a></li>
        </ul>
        <hr>
    </div>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
```

Figure 3.25: Exemplu de bootstrap aplicat

Figure 3.26: Pagina html cu bootstrap

4 Concluzii

Prin implementarea operațiunilor fundamentale CRUD (Create, Read, Update, Delete), aplicația oferă o soluție directă pentru catalogarea artiștilor, clasificarea genurilor muzicale și planificarea evenimentelor live, facilitând astfel administrarea eficientă a informațiilor în domeniul muzical. Această abordare minimală și focalizată permite utilizatorilor să mențină și să acceseze ușor baza de date, făcând această aplicație esențială pentru organizatorii de concerte și entuziaștii muzicii.

5 Bibliografie

References

[1] Oracle. (2023). *The Java EE 7 Tutorial*. https://docs.oracle.com/javaee/7/tutorial/.

[2] MySQL. (2023). *MySQL Documentation*. https://dev.mysql.com/doc/.

[3] Bootstrap. (2023). *Bootstrap Documentation*. https://getbootstrap.com/docs/.

[4] W3Schools. (2023). *SQL Tutorial*. https://www.w3schools.com/sql/.

[5] MDN Web Docs. (2023). *HTML: Hypertext Markup Language*. https://developer.mozilla.org/en-US/docs/Web/HTML.

[6] Codecademy. (2023). *Learn JavaServer Pages*. https://www.codecademy.com/learn/learn-jsp.