

UNIVERSITATEA „POLITEHNICA” din BUCUREȘTI
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

**PROGRAMAREA INTERFETELOR
PENTRU BAZE DE DATE**

Profesor îndrumător:

Pupezescu Valentin

Student:

Mioc Roxana Maria

Grupa 432C

An universitar 2023-2024

Cuprins

1. Cerința	2
2. Descrierea sistemului de gestiune a bazelor de date MySQL	2
3. Tehnologia JSP utilizata in dezvoltarea aplicației	3
4. Limbajul HTML	3
5. Descrierea aplicației	4
5.1. Baza de date	4
5.2. Diagrama logică a bazei de date	6
5.3. Funcționalitatea aplicației	6
6. Concluzii	15
7. Bibliografie	15

1. Cerința

Creați două aplicații care să conțină o bază de date creată în sistemul de gestiune a bazelor de date MySQL și două interfețe la aceasta (baza de date este comună). La crearea interfețelor se vor folosi două tehnologii (la alegere - ex.: JSP, Hibernate, JPA, .NET, Python etc.). Baza de date va fi compusă din tabelele stabilite în lista de teme. Tot în fișierul cu teme sunt stabilite și asocierile dintre tabele.

Interfețele vor trebui să permită utilizatorului să execute următoarele operații pe toate tabele (inclusiv pe cele de legătura dacă aveți asocieri de tip M:N): vizualizare, adăugare, modificare și ștergere de date. Vizualizarea tabelor de legătură va presupune vizualizarea datelor referite din celelalte tabele.

2. Descrierea sistemului de gestiune a bazelor de date 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), 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 componentă 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]

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. [2]

3. Tehnologia JSP utilizata în dezvoltarea aplicației

Tehnologia Java Server Pages (JSP) este cea mai populara metoda de a crea interfețe Web pentru aplicatiile care ruleaza pe platforma Java, creata de Sun. Ea se bazeaza pe tehnologia numita Java Servlets fiind, de fapt, o completare a acesteia in ideea crearii cat mai facile a paginilor Web dinamice. [3]

Punctul central al tehnologiei o reprezinta asa-numitele pagini JSP care sunt, practic, fisiere text care combina descrieri HTML cu cod Java. Paginile JSP sunt gestionate si accesibile prin intermediul unui server de aplicatii. Acesta primeste cereri venite prin HTTP de la un browser Web. Daca o cerere refera o pagina JSP, serverul prelucreaza local pagina respectiva si, in functie de continutul acesteia, genereaza dinamic o pagina HTML pe care o trimite, ca raspuns, browser-ului. [3]

4. Limbajul HTML

HTML este o formă de marcare orientată către prezentarea documentelor text pe o singura pagină, utilizând un software de redare specializat, numit agent utilizator HTML, cel mai bun exemplu de astfel de software fiind browserul web. [4]

HTML se poate genera direct utilizând tehnologii de codare din partea serverului cum ar fi PHP, JSP sau ASP. Multe aplicații ca sistemele de gestionare a conținutului, wiki-uri și forumuri web generează pagini HTML. [4]

5. Descrierea aplicației

5.1. Baza de date

Am realizat o baza de date in MySQL WORKBENCH , ea conținând 3 tabele: **modeltelefon**, **companie** și **locatiefabrica**.

Pentru tabela **modeltelefon**: am ales ca și cheie primară idModelTelefon, celelalte atribute sunt: Nume, Memorie_Interna, Memorie_RAM, Dimensiune_Ecran și An_Aparitie.




Table Name:

Schema: **telefon**

Charset/Collation:

Engine:

Comments:







Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idModelTelefon	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input 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
 Memorie_Interna	FLOAT	<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
 Memorie_RAM	FLOAT	<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
 Dimensiune_Ecran	FLOAT	<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
 An_Aparitie	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

Figura 5.1.1. Atributele tablei modeltelefon

Pentru tabela **locatiefabrica**: am ales ca și cheie primară idLocatieFabrica , celelalte atribute sunt: Oras și Tara.




Table Name:

Schema: **telefon**

Charset/Collation:

Engine:

Comments:




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 idLocatieFabrica	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 Oras	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
 Tara	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

Figura 5.1.2. Atributele tablei locatiefabrica

Pentru a realiza relația de legatura M:N între tabelele **modeltelefon** și **locatiefabrica** trebuie sa existe o a treia tabelă, numită tabelă de legătură. În cazul de față, această tabelă este tabela **companie**. În această nouă tabelă, attributele ce au fost selectate ca si chei primare pentru tabelele anterioare vor deveni chei straine (FK) pentru tabela de legătură **companie**.

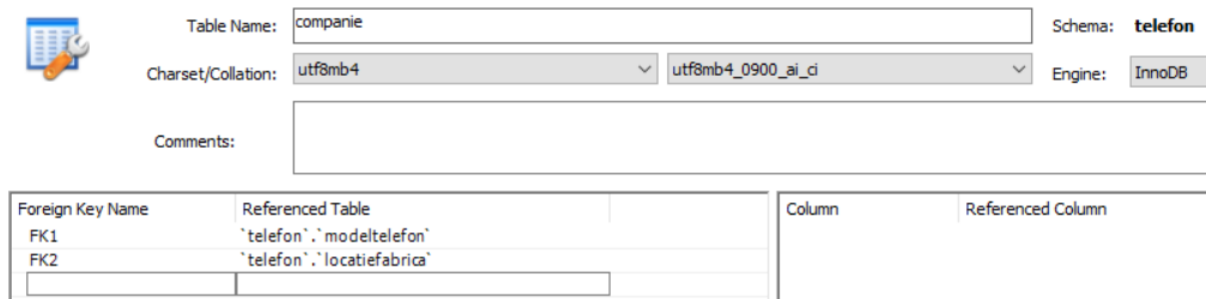


Table Name: Schema: **telefon**

Charset/Collation: Engine: **InnoDB**

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
FK1	'telefon`.`modeltelefon`'		
FK2	'telefon`.`locatiefabrica`'		

Figura 5.1.3. Chei straine (FK) pentru tabela de legătură companie

Pentru tabela **companie**: am ales ca și cheie primară idCompanie, celelate attribute sunt: Nume, Fondator și An_Infiintare.

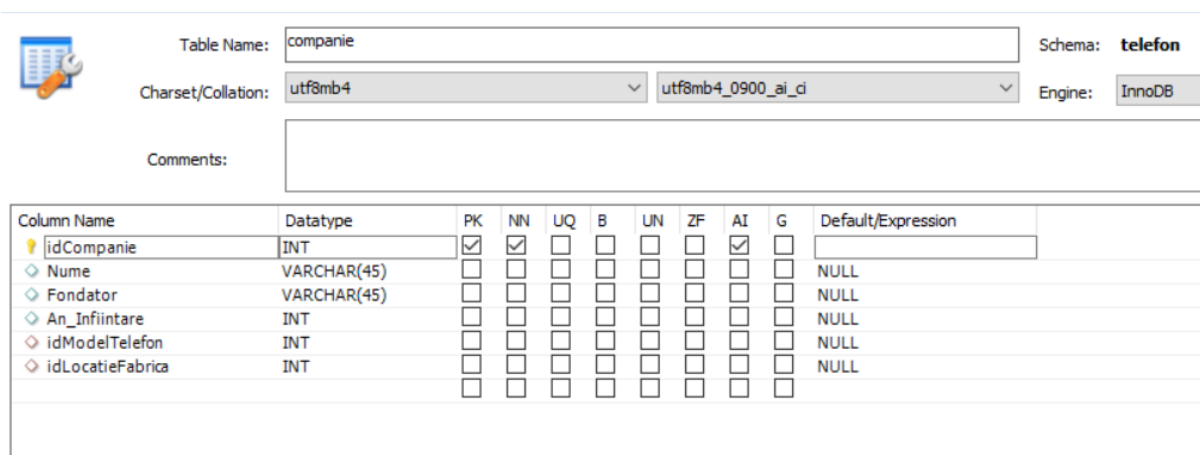


Table Name: Schema: **telefon**

Charset/Collation: Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idCompanie	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input 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
Fondator	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
An_Infiintare	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
idModelTelefon	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
idLocatieFabrica	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

Figura 5.1.4. Attributele tablei companie

5.2. Diagrama logica a bazei de date

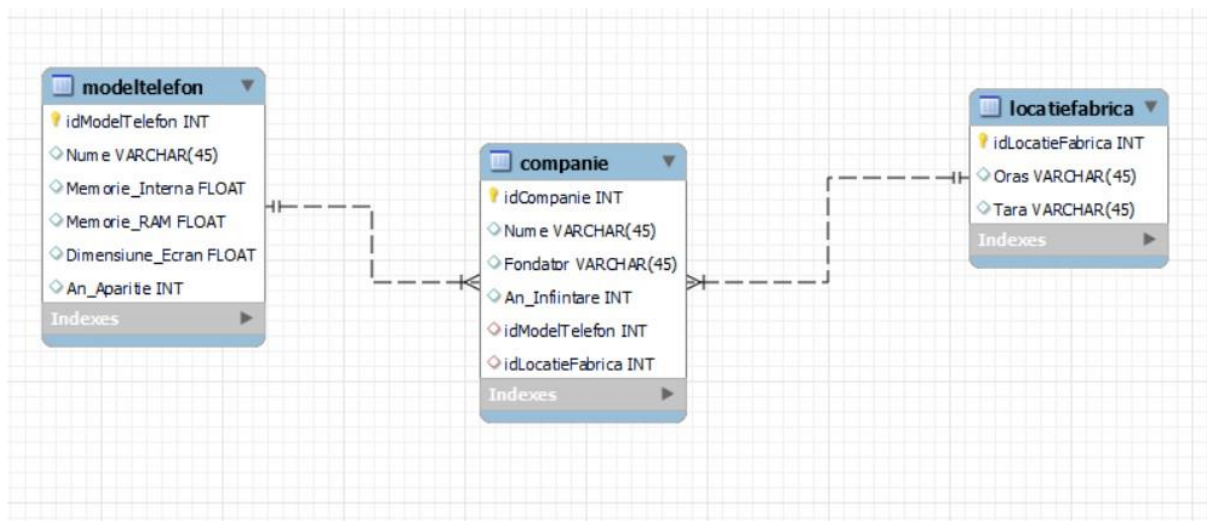


Figura 5.2.1. Diagrama logică a bazei de date

Relațiile de legătură între cele trei tabele:

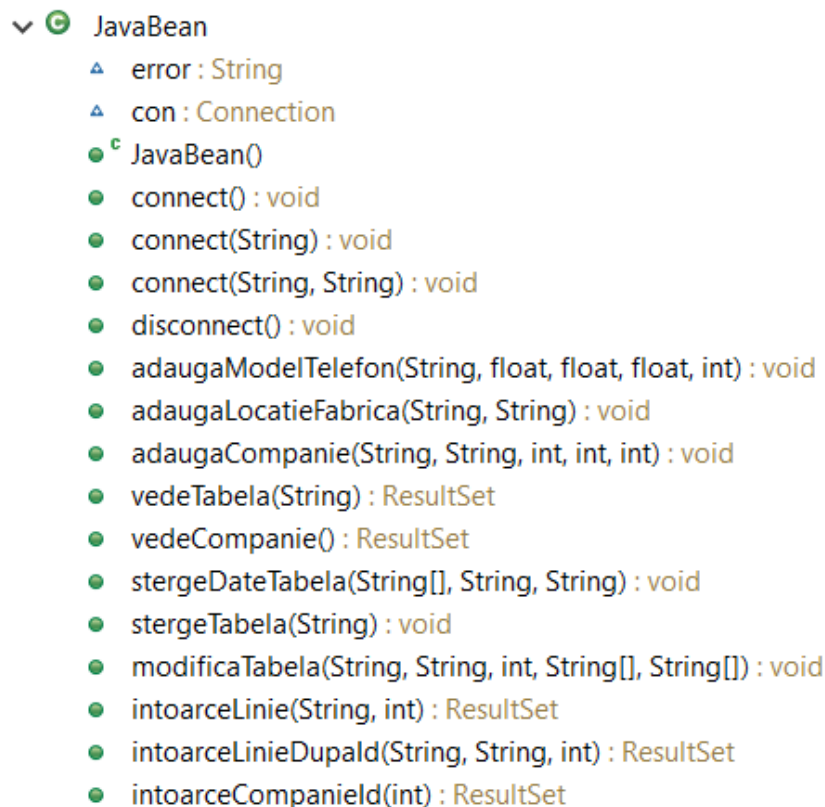
- Între **companie** și **modeltelefon** este o asociere de tipul 1:M
- Între **companie** și **locatiefabrica** este o asociere de tipul 1:N
- Între **modeltelefon** și **locatiefabrica** este o asociere de tipul M:N

5.3. Funcționalitatea aplicației

Proiectul realizat în tehnologia JSP are următoarea structură:

- Un pachet DB ce continue clasa JavaBean, ce are rolul de a oferi toate funcționalitățile principale ale interfeței bazei de date si anume: conectica, operațiunile de afișare, adăugare, modificare, ștergere.
- Folder-ul webapp ce continue toate paginile JSP, ce au rolul de a importa funcționalitățile din clasa JavaBean, fiind conectate între ele si implicit conectate la pachetul DB, ce realizează partea dinamica a proiectului.

Diagrama UML pentru interfața JSP:



După cum putem observa, avem o singură clasă cu mai multe metode care ne permit sa facem toate operațiunile de care avem nevoie: vizualizare tabele (metodele `vedeTabela` si `vedeCompanie`), inserare de date in tabele (metodele `adaugaModelTelefon`, `adaugaLocatieFabrica` si `adaugaCompanie`), ștergere date existente (`stergeDateTabela`, `stergeTabela`), modificare date (`modificaTabela`) dar si metode ce ne permit întoarcerea unei linii folosind id-ul.

Pagina principala a interfeței (**index.html**) făcută în html permite accesul spre paginile tabelor unde putem vizualiza datele, putem adăuga date noi, putem modifica datele existente sau putem șterge ce date dorim.

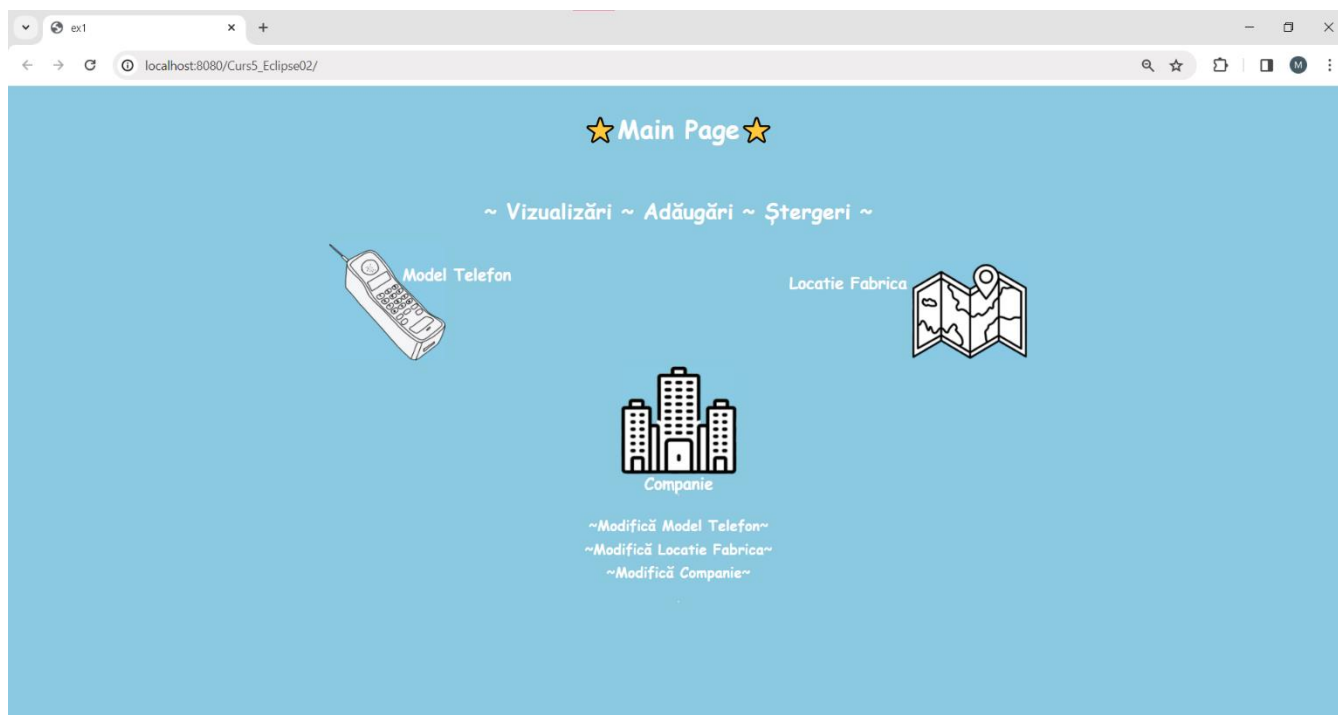


Figura 5.3.1. Pagina principala a interfeței

Din pagina principala, index.html, utilizatorul are posibilitatea de a intra pe 3 pagini aferente tabelor “Model Telefon”, “Locatie Fabrica” și “Companie”, pe fiecare dintre aceste pagini poate vizualiza tabelele din baza de date, poate adauga și șterge date.

Funcționalitatea paginilor este similară pentru cele trei tabele, astfel: pentru a adăuga o înregistrare nouă în tabela (**modeltelefon**, **locatiefabrica** sau **companie**) utilizatorul trebuie să apese pe butonul tabelii corespunzătoare. Se va deschide o pagina în care sunt afișate datele din tabela respectivă, apoi prin apăsarea linkului “Adauga un nou/ o noua....” acesta va fi redirecționat în pagina aferentă adăugării, unde există casete de input.

Pentru funcția de **modificare** a oricărei linii dintr-o tabelă, fiecare are o pagină corespunzătoare.

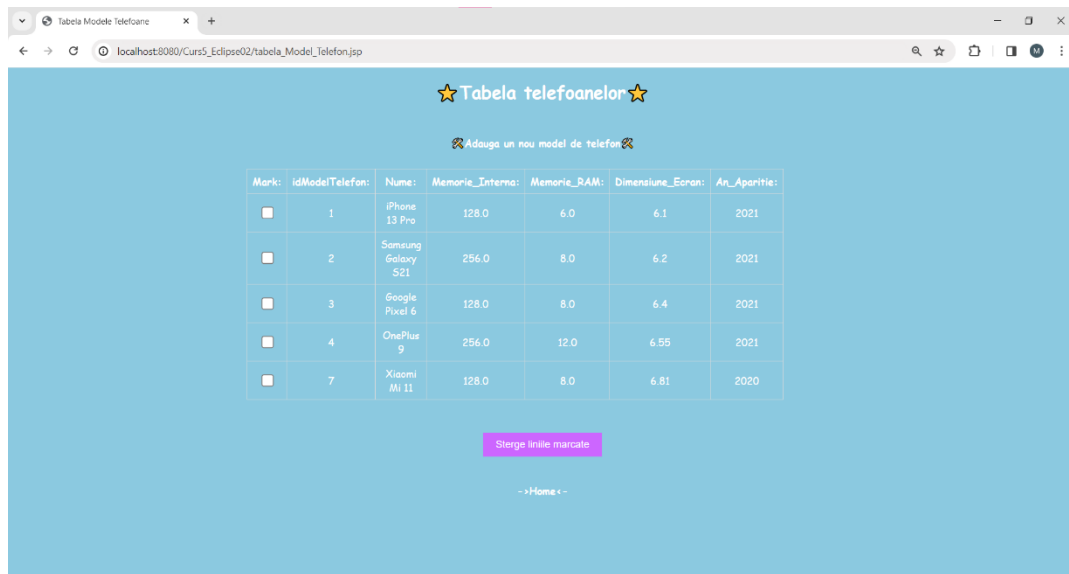


Figura 5.3.2. Tabela telefoanelor

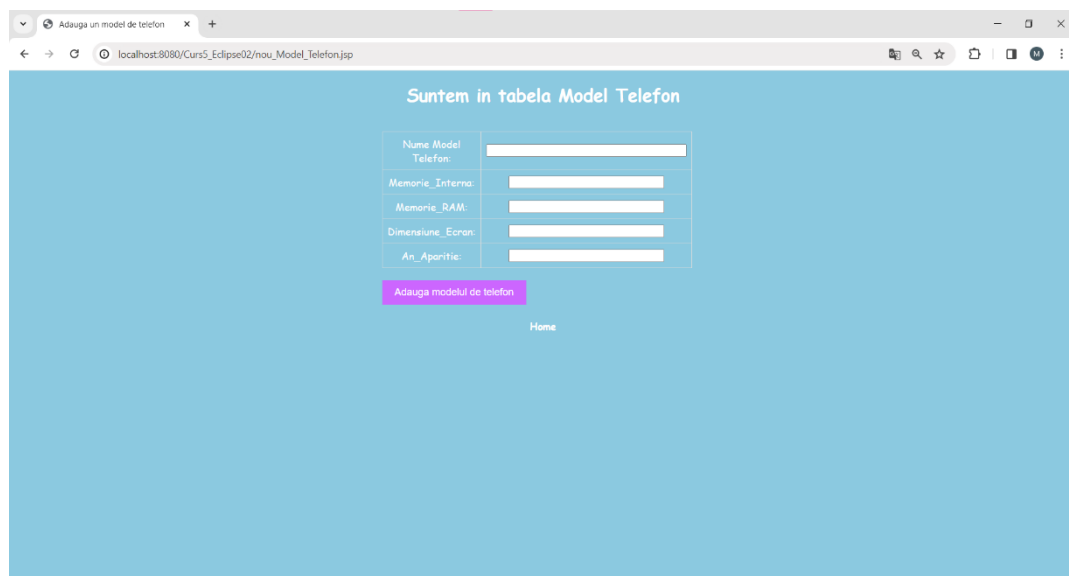


Figura 5.3.3. Adăugare în tabela modeltelefon

Instrucțiunea corespunzătoare **adăugării** unui model de telefon în baza de date este:

```
public void adaugaModelTelefon(String Nume, float Memorie_Interna, float Memorie_RAM, float Dimensiune_Ecran, int An_Aparitie)
    throws SQLException, Exception {
    if (con != null) {
        try {
            // create a prepared SQL statement
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate("insert into modeltelefon(Nume, Memorie_Interna, Memorie_RAM, Dimensiune_Ecran, An_Aparitie) values('" + Nume + "', '" + Memorie_Interna + "', '" +
                Memorie_RAM + "', '" + Dimensiune_Ecran + "', '" + An_Aparitie + "');");

        } 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 adaugaModelTelefon()
```

După cum se poate vedea in imaginea cu codul, se face un insert in tabela **modeltelefon** cu toate câmpurile necesare (nume, memorie interna, memorie RAM, dimensiune ecran, an aparitie).

Instrucțiunea este înconjurată de un block try-catch pentru ca exista posibilitatea sa apară erori, de exemplu sa nu mai existe stocare pe server pentru a adăuga in baza de date sau sa existe duplicate, sa existe alte erori SQL, etc.

Adăugarea in tabela **companie** este similară, dar diferă puțin datorita faptului ca tabela Furnizori este tabela de legătură si trebuie să selectăm produsul si componenta aferente unui furnizor dintr-o lista deja definita:

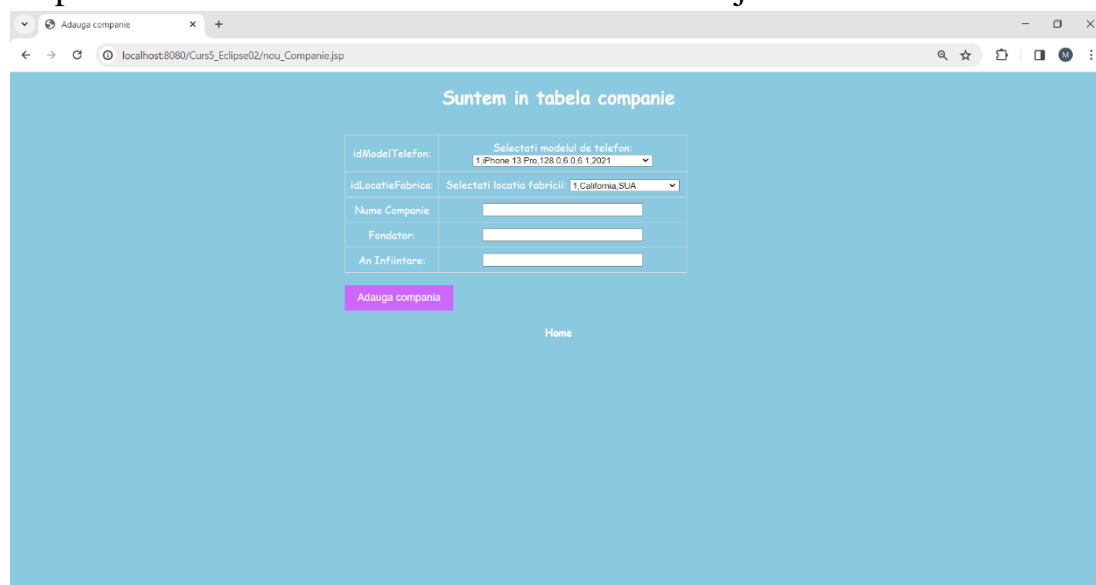


Figura 5.3.4. Pagina pentru adăugarea unei noi companii

Codul corespunzător instrucțiunii de **adăugare** în tabela companie:

```
public void adaugaCompanie(String Nume, String Fondator, int An_Infiintare, int idModelTelefon, int idLocatieFabrica)
    throws SQLException, Exception {
    if (con != null) {
        try {
            // create a prepared SQL statement
            Statement stmt;
            stmt = con.createStatement();
            stmt.executeUpdate("insert into companie(Nume, Fondator, An_Infiintare, idModelTelefon, idLocatieFabrica) values('" + Nume + "', '" + Fondator + "', '" + An_Infiintare + "', '" +
                                idModelTelefon + "', '" + idLocatieFabrica + "');");
        } 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);
    }
}
```

Similar cu tabela produse și în tabela companie se face tot un insert în baza de date cu câmpurile idModelTelefon, idLocatieFabrica, nume, fondator, an infiintare.

După adăugarea cu succes a datelor, pe ecran apare mesajul “Datele au fost adăugate” si un buton ce redirecționează utilizatorul către pagina principală.

Opțiunea de **ștergere** o putem accesa în modul de vizualizare al fiecărei tabele în parte, bifând linia pe care dorim să o ștergem, apăsând apoi pe butonul “Sterge liniile marcate”:

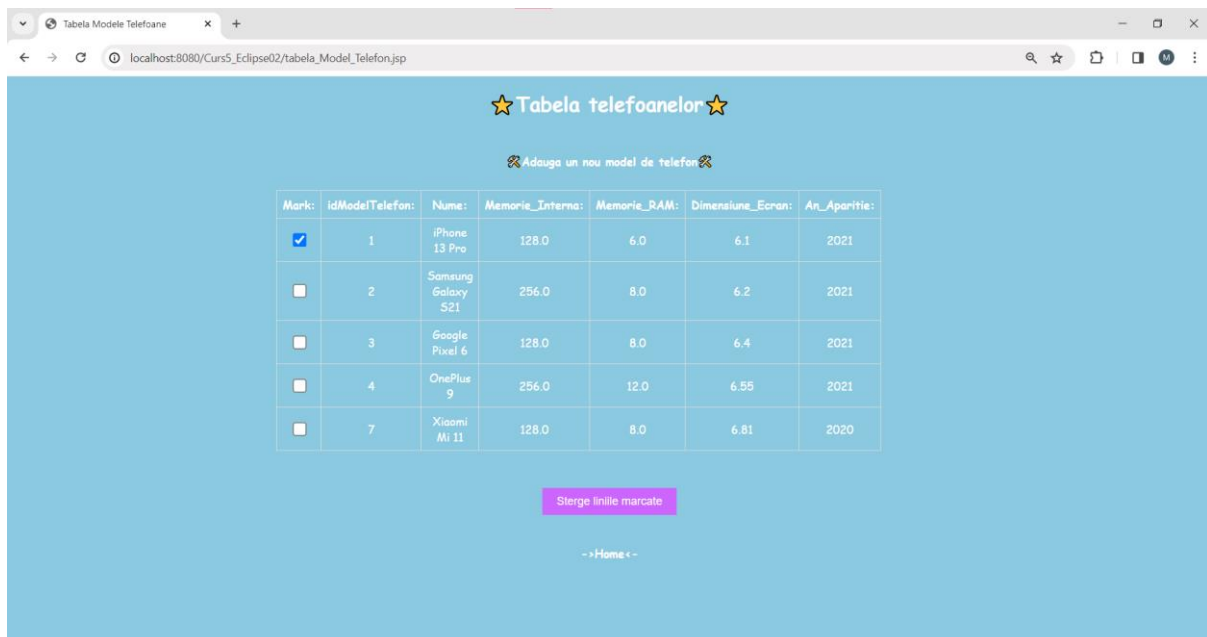


Figura 5.3.5. Ștergerea din tabela modeltelefon

Codul prin care se efectuează **ștergerea** unei linii alcătuiește instrucțiunea DELETE din MySQL, folosindu-se de ID-ul liniei bifate.

```
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()
```

La final va apărea pe ecran mesajul de confirmare a efectuării operației și butonul pentru pagina principală.

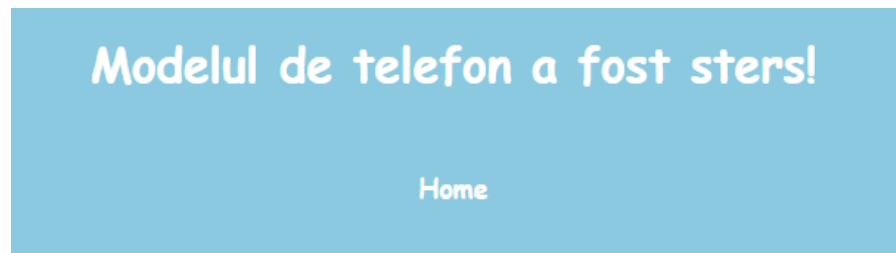


Figura 5.3.6. Confirmare de stergere a datelor

Pentru a efectua **modificări** pe oricare dintre cele 3 tabele avem 3 pagini separate pentru fiecare. Acestea urmăresc același mod de a modifica linia dintr-o tabelă:

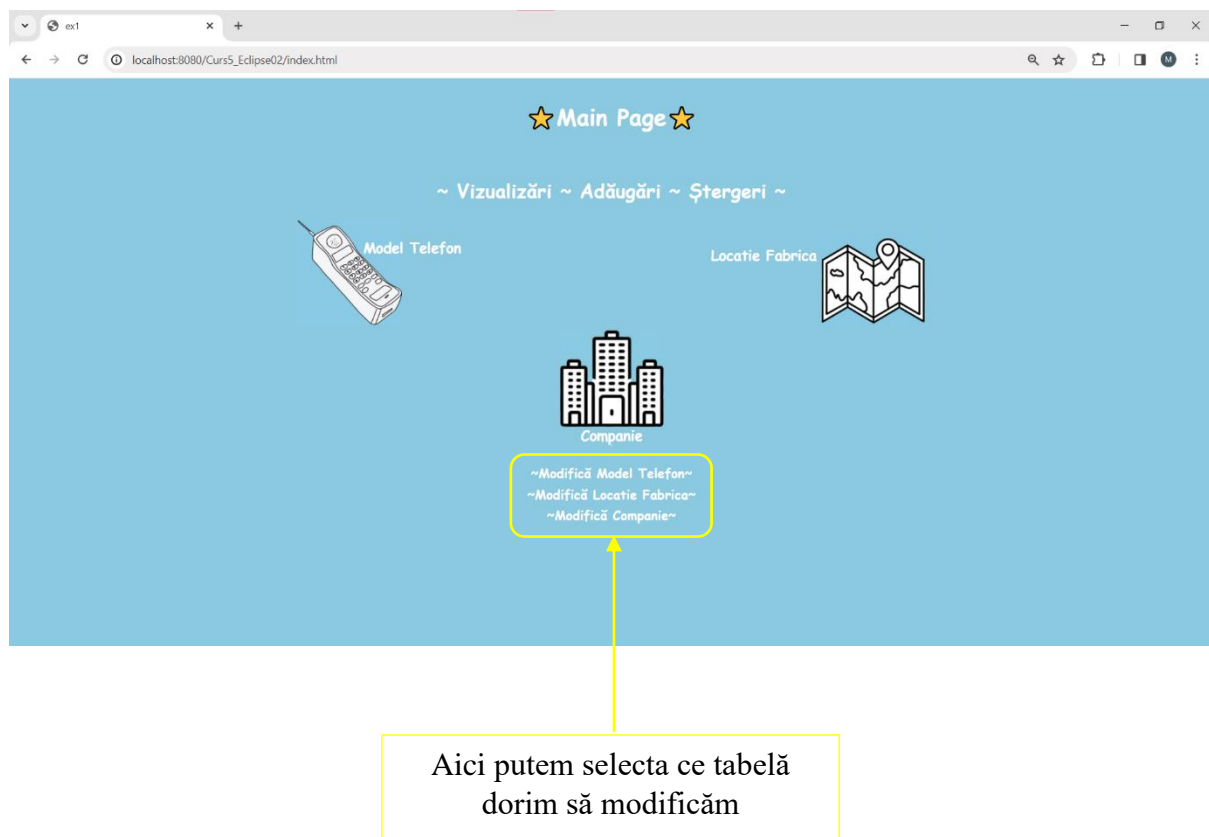


Tabela Model Telefon

✂️ Adauga un nou model de telefon ✂️

Mark:	idModelTelefon:	Nume:	Memorie_Interna:	Memorie_RAM:	Dimensiune_Ecran:	An_Aparitie:
<input type="checkbox"/>	1	iPhone 13 Pro	128.0	6.0	6.1	2021
<input type="checkbox"/>	2	Samsung Galaxy S21	256.0	8.0	6.2	2021
<input type="checkbox"/>	3	Google Pixel 6	128.0	8.0	6.4	2021
<input type="checkbox"/>	4	OnePlus 9	256.0	12.0	6.55	2021
<input type="checkbox"/>	7	Xiaomi Mi 11	128.0	8.0	6.81	2020
<input checked="" type="checkbox"/>	17	sss	128.0	2.0	6.4	2020

Modifica linia

->Home<-

După ce am selectat linia dorită apăsăm pe butonul “Modifica linia”

(În fiecare tabelă de modificare putem să și adăugăm un nou telefon, o nouă locație sau companie)

Figura 5.3.7. Selectarea liniei ce se dorește a fi modificată

Figura 5.3.8. Casete de input unde introducem noile valori.

Tabela Model Telefon

Adauga un nou model de telefon

idModelTelefon:	<input type="text" value="17"/>
Nume:	<input type="text" value="sss"/>
Memorie_Interna:	<input type="text" value="128.0"/>
Memorie_RAM:	<input type="text" value="2.0"/>
Dimensiune_Ecran:	<input type="text" value="6.4"/>
An_Aparitie:	<input type="text" value="2020"/>

Modifica linia

Home

Adauga un nou model de telefon

Modificarile au fost efectuate!

Home

Figura 5.3.9. Mesajul ce confirmă modificările.

Codul corespunzător **modificării** unei 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()
```

După cum se poate vedea în imaginea de mai sus metoda de **modificare** poate fi apelată pentru oricare din tabele. Variabila de tip string tabela parametrizează tabela, la fel și vectorii de tip string campuri și valori, prin aceștia se pot modifica un număr variabil de câmpuri în funcție de tabela dată. Pentru asta se folosește un for care parcurge tot vectorul cu câmpuri și în interiorul acestei bucle vom concatena câmpurile și valorile ce se doresc a fi modificate într-un string cu numele temp. Acest string va fi folosit la instrucțiunea SQL de update pentru efectuarea modificărilor.

6. Concluzii

În concluzie acest curs ne-a arătat că putem face destul de ușor o interfață prietenoasă ce permite gestionarea unor date dintr-o bază de date MySQL chiar și de către persoane care nu au cunoștințe legate de SQL sau de limbaje de programare. Acest lucru este important, pentru că în prezent bazele de date sunt utilizate practic peste tot, fiecare companie, întreprindere, mai ales cu caracter de producere, comercializare are nevoie și la sigur implementează în sistemul lor o bază de date.

7. Bibliografie

- [1] <https://www.nav.ro/blog/ce-este-mysql/>
- [2] <https://ro.wikipedia.org/wiki/MySQL>
- [3] <https://labs.cs.upt.ro/labs/sprc/html/jsp.html>
- [4] [https://ro.wikipedia.org/wiki/HyperText Markup Language](https://ro.wikipedia.org/wiki/HyperText_Markup_Language)