

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 Hibernate utilizată 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 .....	17
7. Bibliografie .....	17

## 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 Hibernate utilizată în dezvoltarea aplicației

Hibernate ORM este un object-relational mapping framework pentru limbajul Java. Acesta oferă un framework pentru maparea a unui model de domeniu orientat pe obiect la o bază de date relațională. Hibernate rezolvă probleme de impedanță nepotrivire la obiect-relaționale prin înlocuirea bazei de date directă, baza de date persistentă accesează obiectele de nivel înalt cu funcții de manipulare. [3]

Caracteristica principală a Hibernate-ului este maparea din clase Java la tabele de baze de date și maparea de la Java data types la SQL data types. Hibernate oferă, de asemenea interogarea de date și facilități de recuperare. [3]

### 4. Limbajul HTML

HTML este o formă de marcare orientată către prezentarea documentelor text pe o singură 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, celelate atribute sunt: Nume, Memorie\_Interna, Memorie\_RAM, Dimensiune\_Ecran și An\_Aparitie.

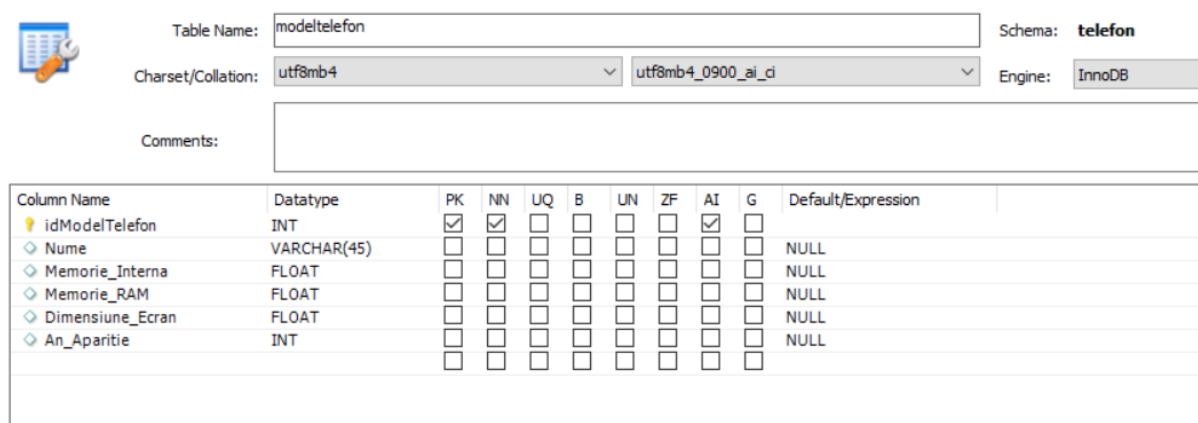


Table Name:  Schema: **telefon**

Charset/Collation:   Engine: **InnoDB**

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 , celelate atribute sunt: Oras și Tara.

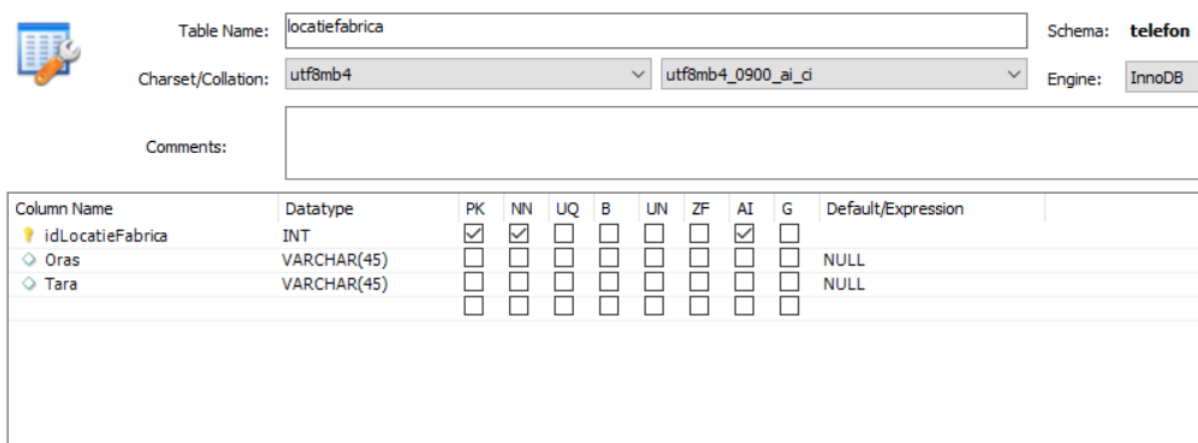


Table Name:  Schema: **telefon**

Charset/Collation:   Engine: **InnoDB**

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 legătura 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**.

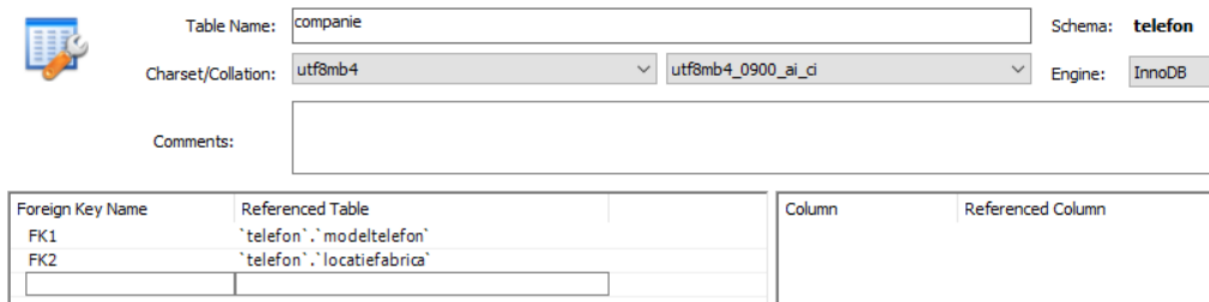


Table Name:  Schema: **telefon**

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**, celelalte attribute sunt: **Nume**, **Fondator** și **An\_Infiintare**.

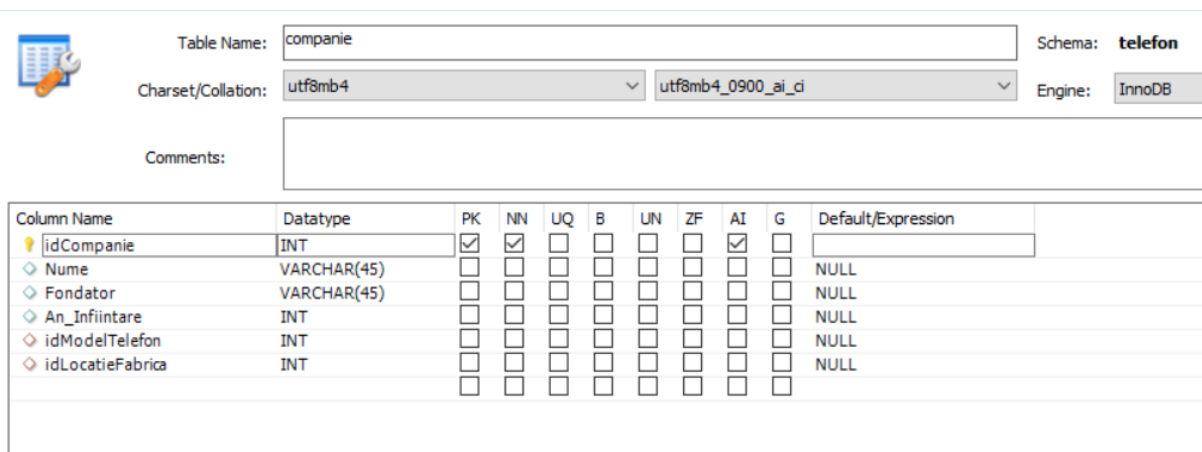


Table Name:  Schema: **telefon**

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

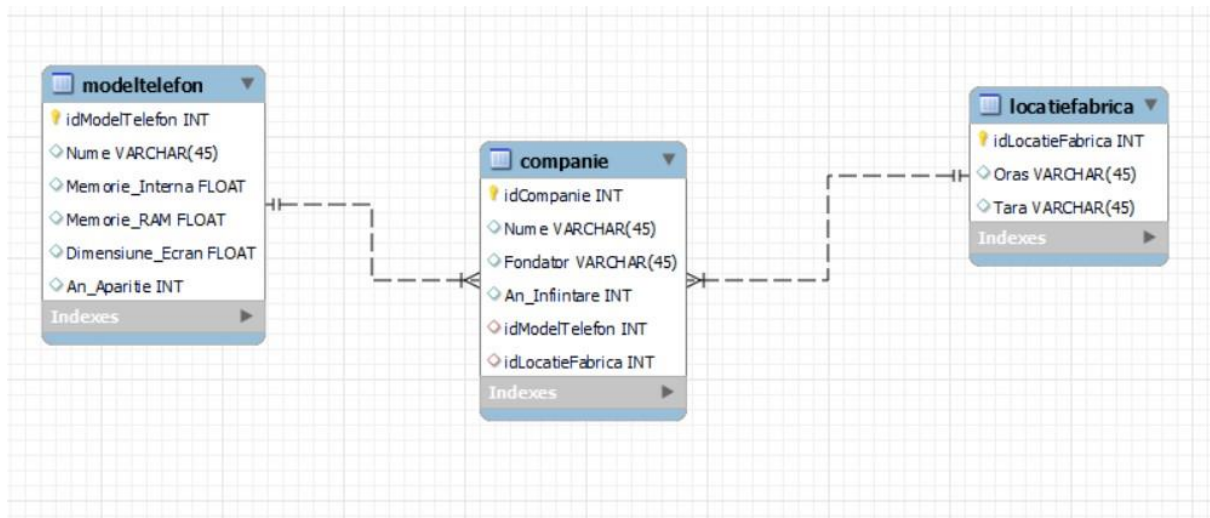


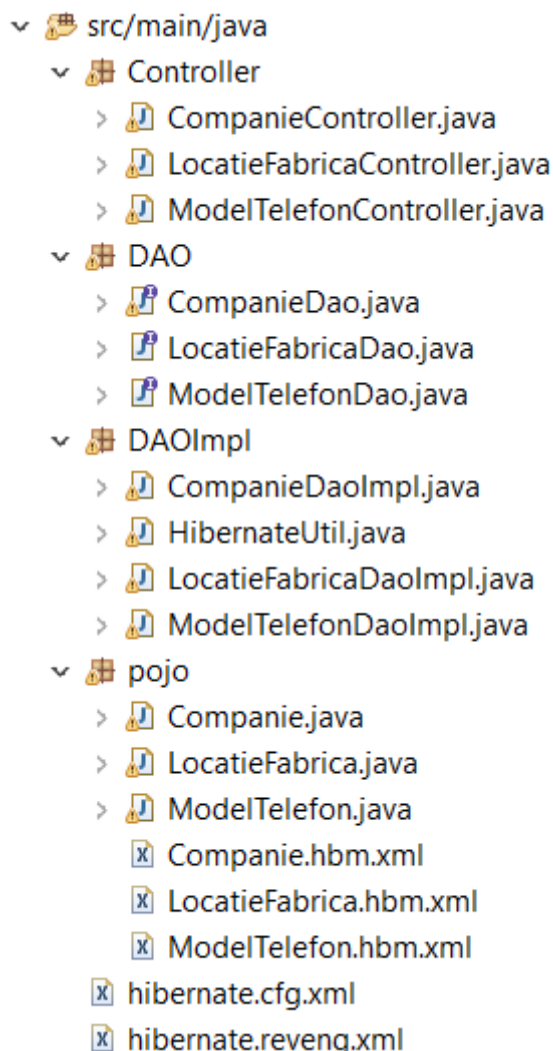
Figura 5.2.1. Diagrama logică a bazei de date

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

Arhitectura proiectului este de tip Model View Controller (MVC).



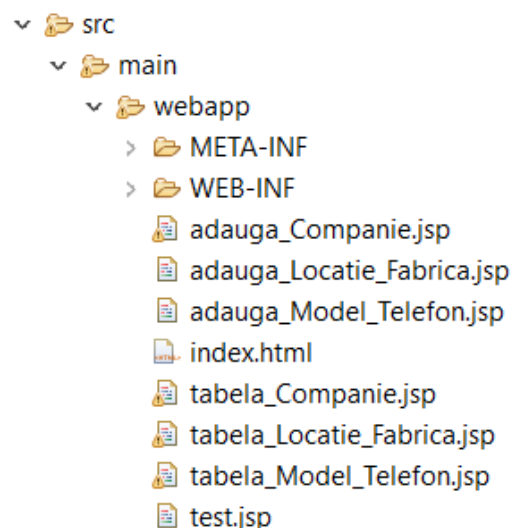
Așadar putem spune ca proiectul e constituit din:

- Partea de model: identificată în POJO, unde se face maparea. În clasele din acest package, cele mai importante elemente sunt funcțiile de GET și SET. Când se fac operații pe obiectele din clasa, modificările se propagă în baza de date.

- Partea de funcționalitate: este reprezentată de următoarele pachete: Controller, DAO, DaoImpl. Aici se fac operațiile de bază pe obiectele din clase, din acest motiv, această parte este considerată partea dinamică a proiectului. În DAO avem interfețe pentru standardizarea denumirilor ce au fost implementate în DAOImpl, unde toate operațiile de aici sunt făcute în cadrul unor tranzacții. Controller-ul este implementat sub firma de SERVLET.

Totodată o altă parte importantă este cea de vizualizare, implementată prin JSP, prin limbajul HTML ce conturează în totalitate front-end-ul proiectului.

JSP este o tehnologie care permite dezvoltatorilor să genereze pagini web dinamice, utilizând fragmente de cod Java, introduse în pagini HTML.





Pagina principala a interfeței (**index.html**) făcută în html permite accesul spre paginile tabelelor unde putem adăuga date noi.

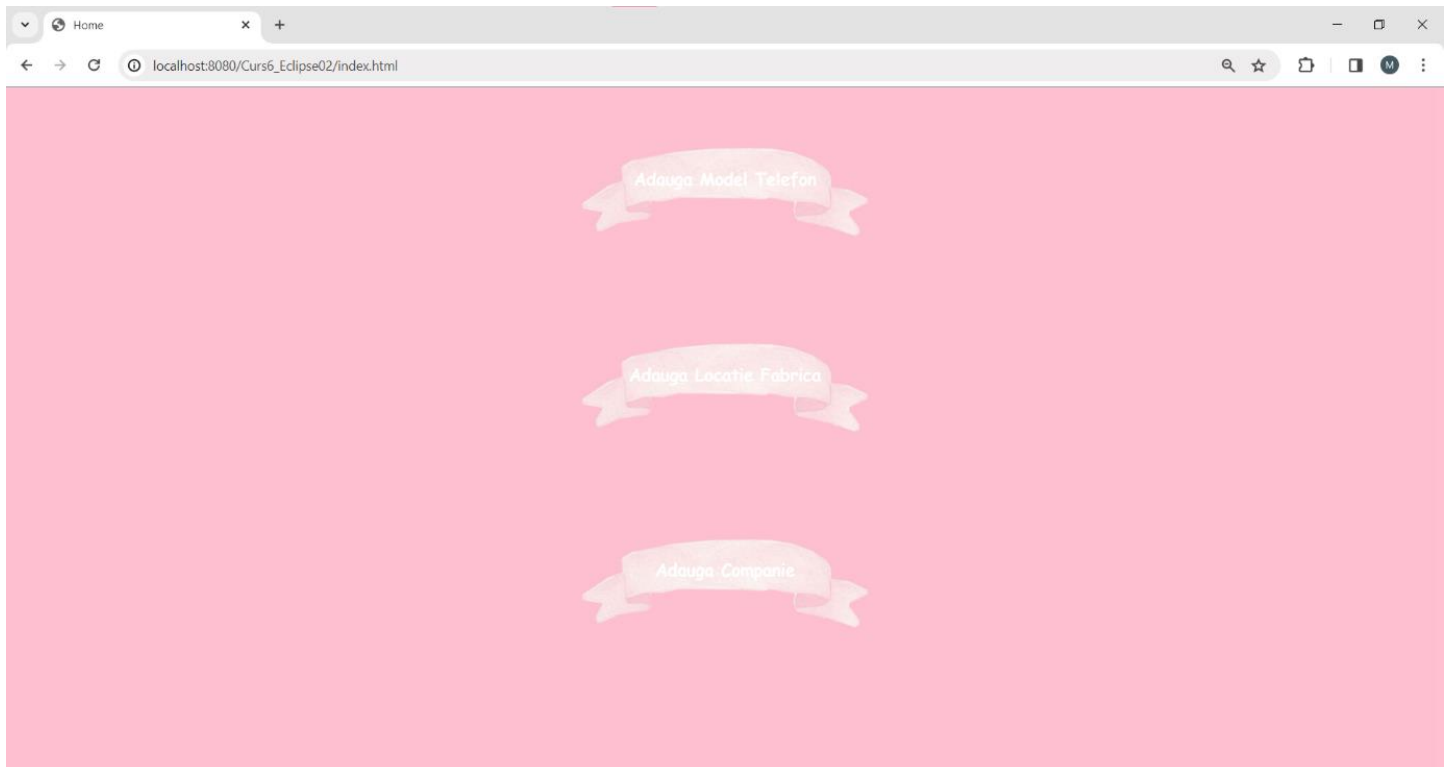


Figura 5.3.1. Pagina principala a interfeței

Din pagina principala, index.html, utilizatorul are posibilitatea de a intra pe 3 pagini aferente tabelelor “Model Telefon”, “Locatie Fabrica” și “Companie”, pe fiecare dintre aceste pagini poate vizualiza tabelele din baza de date, poate adăuga, modifica 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 de modificare corespunzător. Se va deschide o pagina în care sunt afișate datele din tabelele corespunzătoare.



Figura 5.3.2. Tabela telefoanelor

Dacă se dorește **adăugarea** unui nou model de telefon, utilizatorul va da click pe adaugă model telefon și se va deschide o pagina cu un formular unde utilizatorul poate complete datele necesare:

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/Curs6\_Eclipse02/ModelTelefonController'. The page has a pink background and is titled 'Adauga un model de telefon'. It contains a form with five input fields for 'Nume Model Telefon', 'Memorie Interna Model Telefon', 'Memorie RAM Model Telefon', 'Dimensiune Ecran Model Telefon', and 'An Aparitie Model Telefon'. Below the form are two buttons: 'Adauga' and 'Afiseaza'. At the bottom is a 'Home' button.

Figura 5.3.3. Adăugare in tabela modeltelefon

Codurile aferente **adăugării** unui model de telefon în baza de date:

```
public class ModelTelefonController extends HttpServlet {

    ModelTelefon telefon = new ModelTelefon();
    ModelTelefonDaoImpl modelTelefonDaoImpl = new ModelTelefonDaoImpl();

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        if (request.getParameter("adaugaModelTelefon") != null) {
            String nume = request.getParameter("nume");
            float memorie_Interna = Float.parseFloat(request.getParameter("memorie_Interna"));
            float memorie_RAM = Float.parseFloat(request.getParameter("memorie_RAM"));
            float dimensiune_Ecran = Float.parseFloat(request.getParameter("dimensiune_Ecran"));
            int an_Aparitie = Integer.parseInt(request.getParameter("an_Aparitie"));
            telefon.setNume(nume);
            telefon.setMemorie_Interna(memorie_Interna);
            telefon.setMemorie_RAM(memorie_RAM);
            telefon.setDimensiune_Ecran(dimensiune_Ecran);
            telefon.setAn_Aparitie(an_Aparitie);
            modelTelefonDaoImpl.adaugaModelTelefon(telefon);
            RequestDispatcher rd = request.getRequestDispatcher("adauga_Model_Telefon.jsp");
            rd.forward(request, response);
        }
    }
}

public class ModelTelefonDaoImpl implements ModelTelefonDao{

    public void adaugaModelTelefon(ModelTelefon tel) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = session.beginTransaction();
        session.save(tel);
        transaction.commit();
        session.close();
    }
}
```

În prima poza avem un cod din clasa ModelTelefonController care ne permite preluarea datelor din formularul html folosind metoda request.getParameter(). Folosind aceste date cream un obiect de tip ModelTelefon și îl adăugăm în baza de date folosind clasa ModelTelefonDaoImpl. În această clasă folosind metoda adaugaModelTelefon și câmpul static session din HibernateUtil putem adăuga un model de telefon nou (prin session.save(tel)).

Adăugarea în tabela **locatiefabrica** este complet analoagă, se folosesc clasele LocatieFabricaController și LocatieFabricaDaoImpl pentru adăugarea unei noi locații în baza de date.

Adăugarea în tabela **companie** este similară, dar diferă puțin datorită faptului că tabela **companie** este tabela de legătura și trebuie să selectăm modelul de telefon și locația fabricii aferente unei companii dintr-o listă deja definită:

Figura 5.3.4. Pagina pentru adăugarea unei noi companii

Codurile aferente **adăugării** unei companii în baza de date:

```
public class CompanieController extends HttpServlet {

    Companie companie = new Companie();
    CompanieDaoImpl companieDaoImpl = new CompanieDaoImpl();

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        if (request.getParameter("adaugaCompanie") != null) {
            Integer idModelTelefon = java.lang.Integer.parseInt(request.getParameter("idModelTelefon"));
            Integer idLocatieFabrica = java.lang.Integer.parseInt(request.getParameter("idLocatieFabrica"));
            Session session = HibernateUtil.getSessionFactory().openSession();
            ModelTelefon modelTelefon = (ModelTelefon) session.get(ModelTelefon.class, idModelTelefon);
            LocatieFabrica locatieFabrica = (LocatieFabrica) session.get(LocatieFabrica.class, idLocatieFabrica);

            String nume = request.getParameter("nume");
            String fondator = request.getParameter("fondator");
            int an_Infiintare = java.lang.Integer.parseInt(request.getParameter("an_Infiintare"));

            companie.setModelTelefon(modelTelefon);
            companie.setLocatieFabrica(locatieFabrica);
            companie.setNume(nume);
            companie.setFondator(fondator);
            companie.setAn_Infiintare(an_Infiintare);

            companieDaoImpl.adaugaCompanie(companie);
            RequestDispatcher rd = request.getRequestDispatcher("adauga_Companie.jsp");
            rd.forward(request, response);
        }
    }
}
```

```

public class CompanieDaoImpl implements CompanieDao{

    public void adaugaCompanie(Companie comp) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = session.beginTransaction();
        session.save(comp);
        transaction.commit();
        session.close();
    }

    public List<Companie> afiseazaCompanie() {
        List<Companie> listaCompanie = new ArrayList();
        Session session = HibernateUtil.getSessionFactory().openSession();
        org.hibernate.Query query = session.createQuery("FROM Companie");
        listaCompanie = query.list();

        return listaCompanie;
    }
}

```

Datorita faptului ca tabela **companie** este o tabelă de legătura în funcția doPost din CompanieController a trebuit să facem rost de modelul de telefon si locația fabricii asociate unei companii folosindu-ne de metoda session.get() și de ID-ul modelului de telefon, respectiv locației fabricii.

În cea de-a doua imagine putem vedea cum adăugam compania în tabela companie folosindu-ne de metoda session.save(comp).

Pentru a **șterge** câmpuri din tabele bifam chenarul de lângă Sterge și apoi selectăm id-ul clientului pe care dorim să-l ștergem și apăsăm pe tasta “Sterge”:

The screenshot shows a web browser window with the URL `localhost:8080/Curs6_Eclipse02/ModelTelefonController`. The page has a pink background and is titled "Tabela Model Telefon:". It contains a table with 6 columns: IdModelTelefon, Nume, Memorie Interna, Memorie RAM, Dimensiune Ecran, and An Aparitie. The table lists 7 phone models, with the last one having an ID of 17 and the name "sssa". Below the table, there is a "Modifica:" label with two radio buttons: "Modifica" (unchecked) and "Sterge" (checked). A dropdown menu shows the selected ID "17". Below this are five input fields for modifying the phone's details: "Modifica Nume:", "Modifica Memorie Interna:", "Modifica Memorie RAM:", "Modifica Dimensiune Ecran:", and "Modifica An Aparitie:". At the bottom, there are two buttons: "Sterge" (highlighted in blue) and "Home".

IdModelTelefon	Nume	Memorie Interna	Memorie RAM	Dimensiune Ecran	An Aparitie
1	iPhone 13 Pro	128.0	6.0	6.1	2021
2	Samsung Galaxy S21	256.0	8.0	6.2	2021
3	Google Pixel 6	128.0	8.0	6.4	2021
4	OnePlus 9	256.0	12.0	6.55	2021
7	Xiaomi Mi 11	128.0	8.0	6.81	2020
17	sssa	128.0	2.0	6.4	2020

Figura 5.3.5. Ștergerea din tabela modeltelefon

## Codurile aferente ștergerii din tabela **modeltelefon**:

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    if (request.getParameter("afiseazaModelTelefon") != null) {
        List<ModelTelefon> listaModelTelefon = new ArrayList();
        listaModelTelefon = modelTelefonDaoImpl.afiseazaModelTelefon();
        request.setAttribute("listaModelTelefon", listaModelTelefon);
        RequestDispatcher rd = request.getRequestDispatcher("tabela_Model_Telefon.jsp");
        rd.forward(request, response);
    }

    if (request.getParameter("modificaModelTelefon") != null) {
        int id1 = Integer.parseInt(request.getParameter("idModelTelefon"));
        String nume = request.getParameter("nume");
        float memorie_Interna = Float.parseFloat(request.getParameter("memorie_Interna"));
        float memorie_RAM = Float.parseFloat(request.getParameter("memorie_RAM"));
        float dimensiune_Ecran = Float.parseFloat(request.getParameter("dimensiune_Ecran"));
        int an_Aparitie = Integer.parseInt(request.getParameter("an_Aparitie"));
        modelTelefonDaoImpl.modificaModelTelefon(id1, nume, memorie_Interna, memorie_RAM, dimensiune_Ecran, an_Aparitie);
        RequestDispatcher rd = request.getRequestDispatcher("adauga_Model_Telefon.jsp");
        rd.forward(request, response);
    }

    if (request.getParameter("stergeModelTelefon") != null) {
        int id2 = Integer.parseInt(request.getParameter("idModelTelefon"));
        telefon.setIdModelTelefon(id2);
        modelTelefonDaoImpl.stergeModelTelefon(telefon);
        RequestDispatcher rd = request.getRequestDispatcher("adauga_Model_Telefon.jsp");
        rd.forward(request, response);
    }
}

public void stergeModelTelefon(ModelTelefon modeltelefon) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = session.beginTransaction();
    session.delete(modeltelefon);
    transaction.commit();
    session.close();
}
```

Dacă utilizatorul dorește **ștergerea** unui model de telefon, se intra în if-ul aferent ștergerii din metoda `doPost`. În acest if se transforma id-ul din string în int (deoarece este preluat de tip string din formular), apoi se șterge modelul de telefon cu id-ul respective prin apelarea metode `stergeModelTelefon`. În această metodă se folosește câmpul static `session` din tabela `HibernateUtil` pentru a efectua ștergerea cu metoda `session.delete(modeltelefon)`.

Codurile aferente ștergerilor sunt analoage celor 3 tabele, diferă doar numele obiectelor utilizate, în rest metodele sunt similare.

Pentru a **modifica** câmpuri din tabele bifăm chenarul de lângă Modifica și apoi selectăm id-ul clientului pe care dorim să-l modificăm, iar la sfârșit după ce am făcut modificările dorite apăsăm pe tasta “Modifica”:

The screenshot shows a web browser window with the URL `localhost:8080/Curs6_Eclipse02/ModelTelefonController`. The page has a pink background and is titled "Tabela Model Telefon:". It contains a table with 6 columns: IdModelTelefon, Nume, Memorie Interna, Memorie RAM, Dimensiune Ecran, and An Aparitie. The table lists 7 phones. Below the table, there are two checkboxes: "Modifica" (checked) and "Sterge" (unchecked). A dropdown menu shows the selected ID "17". Below the dropdown are five input fields for editing the phone's details: "Modifica Nume", "Modifica Memorie Interna", "Modifica Memorie RAM", "Modifica Dimensiune\_Ecran", and "Modifica An Aparitie". At the bottom, there are two buttons: "Modifica" and "Home".

IdModelTelefon	Nume	Memorie Interna	Memorie RAM	Dimensiune Ecran	An Aparitie
1	iPhone 13 Pro	128.0	6.0	6.1	2021
2	Samsung Galaxy S21	256.0	8.0	6.2	2021
3	Google Pixel 6	128.0	8.0	6.4	2021
4	OnePlus 9	256.0	12.0	6.55	2021
7	Xiaomi Mi 11	128.0	8.0	6.81	2020
17	sssa	128.0	2.0	6.4	2020

Figura 5.3.6. Formularul pentru modificare din tabela modeltelefon

Codurile pentru modificarea în tabela `modeltelefon`:

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    if (request.getParameter("afiseazaModelTelefon") != null) {
        List<ModelTelefon> listaModelTelefon = new ArrayList();
        listaModelTelefon = modelTelefonDaoImpl.afiseazaModelTelefon();
        request.setAttribute("listaModelTelefon", listaModelTelefon);
        RequestDispatcher rd = request.getRequestDispatcher("tabela_Model_Telefon.jsp");
        rd.forward(request, response);
    }

    if (request.getParameter("modificaModelTelefon") != null) {
        int id1 = Integer.parseInt(request.getParameter("idModelTelefon"));
        String nume = request.getParameter("nume");
        float memorie_Interna = Float.parseFloat(request.getParameter("memorie_Interna"));
        float memorie_RAM = Float.parseFloat(request.getParameter("memorie_RAM"));
        float dimensiune_Ecran = Float.parseFloat(request.getParameter("dimensiune_Ecran"));
        int an_Aparitie = Integer.parseInt(request.getParameter("an_Aparitie"));
        modelTelefonDaoImpl.modificaModelTelefon(id1, nume, memorie_Interna, memorie_RAM, dimensiune_Ecran, an_Aparitie);
        RequestDispatcher rd = request.getRequestDispatcher("adauga_Model_Telefon.jsp");
        rd.forward(request, response);
    }

    if (request.getParameter("stergeModelTelefon") != null) {
        int id2 = Integer.parseInt(request.getParameter("idModelTelefon"));
        telefon.setIdModelTelefon(id2);
        modelTelefonDaoImpl.stergeModelTelefon(telefon);
        RequestDispatcher rd = request.getRequestDispatcher("adauga_Model_Telefon.jsp");
        rd.forward(request, response);
    }
}
```



```

public void modificaModelTelefon(int idModelTelefon, String nume, float memorie_Interna, float memorie_RAM, float dimensiune_Ecran, int an_Aparitie) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = session.beginTransaction();
    ModelTelefon detaliimodeltelefon = (ModelTelefon) session.load(ModelTelefon.class, idModelTelefon);
    detaliimodeltelefon.setNume(nume);
    detaliimodeltelefon.setMemorie_Interna(memorie_Interna);
    detaliimodeltelefon.setMemorie_RAM(memorie_RAM);
    detaliimodeltelefon.setDimensiune_Ecran(dimensiune_Ecran);
    detaliimodeltelefon.setAn_Aparitie(an_Aparitie);
    session.update(detaliimodeltelefon);
    transaction.commit();
    session.close();
}

```

Pentru modificare se folosește if-ul corespunzător din metoda doPost, în aceasta se preiau câmpurile ce se doresc a fi modificate din formular și se apelează metoda modificaModelTelefon aferentă obiectului modelTelefonDaoImpl. Aici se preia modelul de telefon existent din baza de date MySQL în obiectul detaliimodeltelefon și se modifică toate câmpurile cu datele din formular, apoi folosind session.update(detaliimodeltelefon) se actualizează în baza de date datele modelului de telefon respective.

Codurile aferente modificărilor sunt similar și principiul de baza este același, sunt mici diferențe datorate numărului diferit de câmpuri din fiecare tabelă și a tipului de date.

Baza de date făcută în MySQL este comuna celor 2 tehnologii, deci o modificare făcută în una din ele va fi vizibilă și în cealaltă.

Pentru a îmbunătăți aspectul paginilor am adăugat elemente de programare HTML, Cascading Style Sheets (CSS) dar și mici bucăți de cod javascript.



## 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] [http://repository.utm.md/bitstream/handle/5014/229/Conf\\_UTM\\_2017\\_I\\_pg211-213.pdf?sequence=1&isAllowed=y](http://repository.utm.md/bitstream/handle/5014/229/Conf_UTM_2017_I_pg211-213.pdf?sequence=1&isAllowed=y)
- [4] [https://ro.wikipedia.org/wiki/HyperText\\_Markup\\_Language](https://ro.wikipedia.org/wiki/HyperText_Markup_Language)