A close-up of a white background

Description automatically generated

**PROGRAMAREA INTERFEȚELOR PENTRU BAZE DE DATE**

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

Student: **Radu Crina**

Grupa: **433B**

**Cuprins**

1. Tema Proiectului
2. Descrierea sistemului de gestiune

a bazelor de date MySQL

1. Tehnologia JSP utilizată în dezvoltarea aplicației
2. Limbajul HTML și utilitatea sa în aplicații
3. Descrierea aplicației
4. Concluzii
5. Bibliografie
6. **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. Se pot utiliza diferite tehnologii: JSP, Hibernate, JPA, .NET, Python+Django, Python+Flask, etc. Interfețele vor trebui să permită utilizatorului să execute urmatoarele operații pe toate tabele: vizualizare, adaugare, modificare și ștergere de date. Vizualizarea tabelelor de legatură va presupune vizualizarea datelor referite din celelalte tabele. Pentru tema individuala primită, am ales 2 tehnologii diferite: JSP și Hibernate. Asocierea pentru tabelele din baza de date este de M:N.

1. **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. Acesta permite stocarea, organizarea și recuperarea informațiilor într-o structură de tabel cu rânduri și coloane. Este compatibil cu majoritatea sistemelor de operare, inclusiv Windows, Linux și macOS.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comanda sau, prin descărcare, de pe internet, o interfață grafică: MySQL Administrator și MySQL Query Browser. Baza de date MySQL este folosită în principal ca mijloc de stocare de date pentru aplicații mari, bazate pe web.

1. **Tehnologia JSP utilizată în dezvoltarea aplicației**

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

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

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

Principala diferentaintre JSP și Servlet este caJSP este un limbaj de scripting al paginii web care poate genera continut web dinamic în timp ce servletul este un program Java care este deja compilat și folosit pentru a crea conținut dinamic de web.

Intr-o aplicatie web obisnuita, clientul solicita o pagina Web de la server și serverul raspunde inapoi cu pagina necesara. Aceste pagini pot fi fie statice, fie dinamice. Continutul unei pagini statice este deja creat. Continutul unei pagini dinamice este creat in timpul rularii. JSP și Servlet sunt două metode în Java pentru a crea pagini web dinamice. JSP este tradus si compilat intr-un servlet de catre containerul web. Pe de alta parte, un servlet este un program Java care este gestionat de containerul web.

1. **Limbajul HTML și utilitatea sa în aplicații**

HTML (HyperText Markup Language) este un limbaj de marcare utilizat pentru crearea de pagini web. Folosind etichete HTML, poți defini structura și conținutul unei pagini web, precum titluri, paragrafe, link-uri, imagini și multe altele. Rezultatul final este o pagină web care poate fi vizualizată în browser-ul web. HTML este esențial pentru crearea de site-uri web și este unul dintre elementele de bază ale tehnologiei web.

1. **Descrierea aplicației**

* **Baza de date**

Tema se bazează pe crearea unei baze de date ce are doua tabele în asociere M:N. Tabelele sunt: clienti și comenzi. Pentru cele doua tabele am ales câteva atribute caracteristice:

Pentru tabela clienti am ales ca și cheie primara idclient. Celelalte atribute sunt nume, adresa, telefon, mail.

A screenshot of a computer

Description automatically generated

***Figura 1.*** *Tabela client*

Pentru tabela comenzi am ales ca și cheie primara idcomanda. Celelalte atribute sunt valoare, tipprodus, cantitate.

A screenshot of a computer

Description automatically generated

***Figura 2****. Tabela comenzi*

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 furnizori. Am ales ca și cheie primara idfurnizor. Celelalte atribute sunt idclient, idcomanda, adresa, tipprodus. 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.

A screenshot of a computer

Description automatically generated

***Figura 3.*** *Tabela furnizori*

* **Diagrama logică a bazei de date**

*A screenshot of a computer

Description automatically generated*

***Figura 4.*** *Diagrama logică a bazei de date*

Această diagramă oferă o reprezentare logică și detaliată a datelor din cele 3 tabele.

Între clienti și furnizori există o asociere de tip 1:N. Între clienti si comenzi exista o asociere de tip M:N. Între comenzi și furnizori există o asociere de tip 1:N.

* **Diagrama implementării fișierelor ce alcătuiesc pagina web**

**A diagram of a computer

Description automatically generated**

***Figura 5.*** *Diagrama implementării fișierelor ce alcătuiesc pagina web*

* **Funcționalitatea aplicației, implementarea funcțiilor**

Toate paginile JSP conțin structure de cod java, ce ofera aplicației comportamentul dinamic. Pentru a utiliza funcțiile din clasa java, trebuie să utilizăm tagurile JSP.

Un exemplu putem regăsii în pagina JSP tabela\_Clienti:

**

***Figura 6.*** *Taguri JSP*

Jb este un obiect instanță la clasa JavaBean, astfel facându-se legătura directă între paginile JSP și funcțiile pe care dorim să le implementăm.

Instrucțiunea property=*"\*"* implementează utilizarea tuturor metodelor din clasa JavaBean.

A computer code with white text

Description automatically generated

***Figura 7.*** *Instructiunea jb.connect()*

Conexiunea se realizează prin intermediul instrucțiunii jb.connect(). Prin intermediul acesteia ne legăm la funcția connect() din clasa JavaBean:

A screen shot of a computer code

Description automatically generated

***Figura 8.*** *Functia connect()*

Obiectul “con” este cel prin intermediul caruia se realizeaza toate operatiile pe baza de date. In functia connect(): se incarca driver-ul de MySQL. DriverManager contine mai multe drivere de conectica, mai multe SGBD-uri.

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

Există și o funcție disconnect() ce realizează close() pe obiectul de conexiune con:

A computer screen shot of a program code

Description automatically generated

***Figura 9.*** *Functia disconnect()*

Cum se realizează afișarea datelor:

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

A screen shot of a computer code

Description automatically generated

***Figura 10.*** *Afisarea datelor*

Se creeaza instructiunea de interogare SQL: 

A screen shot of a computer program

Description automatically generated

***Figura 11.*** *Afisarea datelor*

Incepand cu linia 36 cu structura repetitiva while, se plaseaza ‘’cursorul’’ pe prima linie, si se preia din “idclient” valoarea id-ului ce se leaga de valorile pe care dorim sa le afisam.

Vom obtine urmatoarea pagina:

A screenshot of a computer

Description automatically generated

***Figura 12****. Tabela Clienti*

Stergerea datelor:

Luam drept exemplu stergerea unei comenzi. Asadar ne indreptam atentia catre fisierul sterge\_Comanda.jsp.

A red rectangle with white text

Description automatically generated

***Figura 13.*** *Butonul de stergere*

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

A screen shot of a computer

Description automatically generated

***Figura 14****. Functia getParameterValues()*

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

A screen shot of a computer program

Description automatically generated

***Figura 15.*** *Functia stergeDateTabela()*

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

A blue and white button with black text

Description automatically generated

***Figura 16.*** *Tabela Comenzi Home*

Adaugarea datelor:

Pentru aceasta operatie vom lua ca exemplu tabela de legatura furnizori. Deci ne concentram atentia asupra fisierului nou\_Furnizor.jsp.

A black screen with white text

Description automatically generated

***Figura 17.*** *Folosire getParameter()*

Se declara initial elementele pentru campurile pe care dorim sa le completam. Se preiau parametrii ce actioneaza ca si foreign-keys cu functiile de getParameter() si anume: idcomanda si idclient.

Totodata, am realizat niste casete de tip INPUT pentru introducerea datelor ce se doresc adaugate in baza de date. Acestea vor fi adaugate numai daca in campurile vor fi in totalitate completate, in caz contrar, nu se va adauga nicio data.

A screen shot of a computer code

Description automatically generated

***Figura 18.*** *Introducere date*

Pagina in cadrul careia se adauga datele este:

A screenshot of a computer

Description automatically generated

***Figura 19.*** *Pagina in cadrul careia se adauga datele*

O data apasat butonul “Adauga furnizor”, se va face legatura cu functia nou\_Furnizor() din clasa JavaBean:

A screenshot of a computer program

Description automatically generated

***Figura 20.*** *Functia* nou\_Furnizor()

Modificarea datelor:

A red and white rectangle with text

Description automatically generated

***Figura 21.*** *Modificarea datelor*

Apasand unul dintre butoane vom fi redirectionati catre fisierul JSP corespunzator.

Spre exemplu, daca apasam butonul Modifica Comanda, vom fi redirectionati catre fisireul JSP modifica\_Comanda.

A computer code with white text

Description automatically generated

***Figura 22.*** *Modificarea datelor*

Se realizeaza un formular in cadrul caruia se ruleaza aceleasi instructiuni ca in cazul operatiei de afisare a datelor. Se plaseaza “cursorul” pe prima linie, si se preia din “idcomanda” valoarea id-ului ce se leaga de valorile pe care dorim sa le afisam. Singura diferenta este ca formularul are ca si “action” fisierul JSP m1\_Comanda, ce ia locul unei operatii de adaugare.

A screen shot of a computer screen

Description automatically generated

***Figura 23.*** *Creare ResultSet*

Aici se implementeaza metoda intoarceLinieDupaId() din clasa JavaBean. Aceasta este utilizata pentru a selecta datele ce sunt in legatura cu id-ul comenzii pe care dorim sa o modificam. Utilizam din nou instructiunea specifica MySQL “SELECT \* FROM …. “.

A screen shot of a computer code

Description automatically generated

***Figura 24.*** *Functia intoarceLinieDupaId()*

* **Partea de front-end**

Pentru un aspect mai placut al interfetei, am folosit limbajul HTML combinat cu CSS.

A screenshot of a computer

Description automatically generated

***Figura 25.*** *Aspectul paginii*

Principalul rol al limbajului HTML folosit in cadrul acestei aplicatii este de a plasa componentele specifice design-ului paginii web la anumite coordonate si totodata pentru a seta dimensiunile corespunzatoare pentru div-uri ce compun pagina web.

1. **Concluzii**

Bazele de date sunt utilizate practic peste tot, fiecare companie, intreprindere, avand nevoie de o baza de date. Pentru o utilizare fiabila și corecta a unei baze de date este important sa se urmareasca realizarea unei arhitecturi ce ofera posibilitatea de a separa functionalitatile, si anume partea de back-end de partea de front-end.

Totodata, aplicatia realizata in tehnologia JSP si prezentata in cadrul acestui proiect, numita “Sistem de Administrare al Clienților”, permite utilizatorului sa efectueze operatiile cerute pe baza de date creata in MySQL: afisare, adaugare, modificare si stergere.

1. **Bibliografie**

* Cursuri PIBD
* <https://www.nav.ro/blog/ce-este-mysql/>
* <https://ro.wikipedia.org/>
* <https://web.ceiti.md/lesson.php?id=1>