UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
DISCIPLINA: Baze de Date

Atelier de haine pentru copii

Profesor Coordonator,
Mironeanu Cătălin

Nume student: Jardă Maria-Elisabeta
Grupa student: 1309B

An universitar 2020-2021

Cuprins

Introducere

Titlu proiect : Atelier de haine pentru copii

Atelierul de haine pentru copii este o aplicație care își propune simularea interacțiunii din cadrul unui atelier ce confecționează haine pentru copii și găsirea unor soluții practice de administrare a atelierului. Baza de date conține informații despre angajați, departamente precum și date cu caracter confidențial pentru angajați ,produse vestimentare, mărimi și materialele folosite. Responsabilitățile sunt impartite croitorilor,manager-ilor si designer-ului .

Aplicatia a fost gandita pentru gestionarea atelierului ce isi propune o buna organizare a angajatilor in functie de activitatile fiecaruia,dar si gestionarea materialului ,cat si a bugetului pentru achizitionarea lui.

Aplicația a fost dezvoltată utilizând pe partea de front-end limbajul de programare Java, iar în partea de back-end a fost folosită o bază de date Oracle.

Baza de date contine informatii despre: designer (el se va ocupa de desenarea hainutelor si crearea sabloanelor),croitori (vor croi produsele),manager (el se va ocupa de procurarea materialelor in functie de bugetul alocat pentru fiecare tip de material). Informatiile si datele confidentiale despre acestia se gasesc in tabelele ANGAJAT,CONFIDENTIAL,DEPARTAMENT,CROITOR SI MANAGER

Aplicatia se va ocupa si de: produse (rochita , fusta , camasa, pantalon, sacou, palton, pijama, tricou, hanorac, salopeta) .Acestea sunt impartite pe: categorii de varsta in tabela PRODUS(GIRL->G,BABY GIRL->BG,BOY->B,BABY BOY->BB), si marimi in tabela MARIME

[GIRL si BOY: 2-3 ANI(98 cm), 3-4 ANI(104 cm) ,4-5 ANI(110 cm) , 6 ANI(116 cm), 7 ANI(122 cm)]

[BABY GIRL si BABY BOY:1-3 LUNI(62 cm), 3-6 LUNI(68 cm), 6-9 LUNI(74 cm), 9-12 LUNI(80 cm),12-18 LUNI(86 cm),18-24 LUNI(92 cm)]

Se va tine cont si de tipul materialului in tabela MATERIAL(bumbac,bumbac structurat-> bumbac_s, bumbac moale-> bumbac_m, in, stofa), dar si de culori in tabela MATERIAL (alb,roz,albastru,mustar,etc.).

Descrierea functionala a aplicatiei

Principalele funcţii care se pot întâlni într-un atelier sunt:

- ✓ Evidenta angajatilor
- ✓ Evidenta procurarii de material si gestionarea acestuia
- ✓ Evidenta produselor vestimentare realizate

Capitolul 1. Structura și inter-relaționarea tabelelor

Tabelele din Figura 1.1 simulează interacțiunea ce se stabilește în cadrul unei ferme de animale.

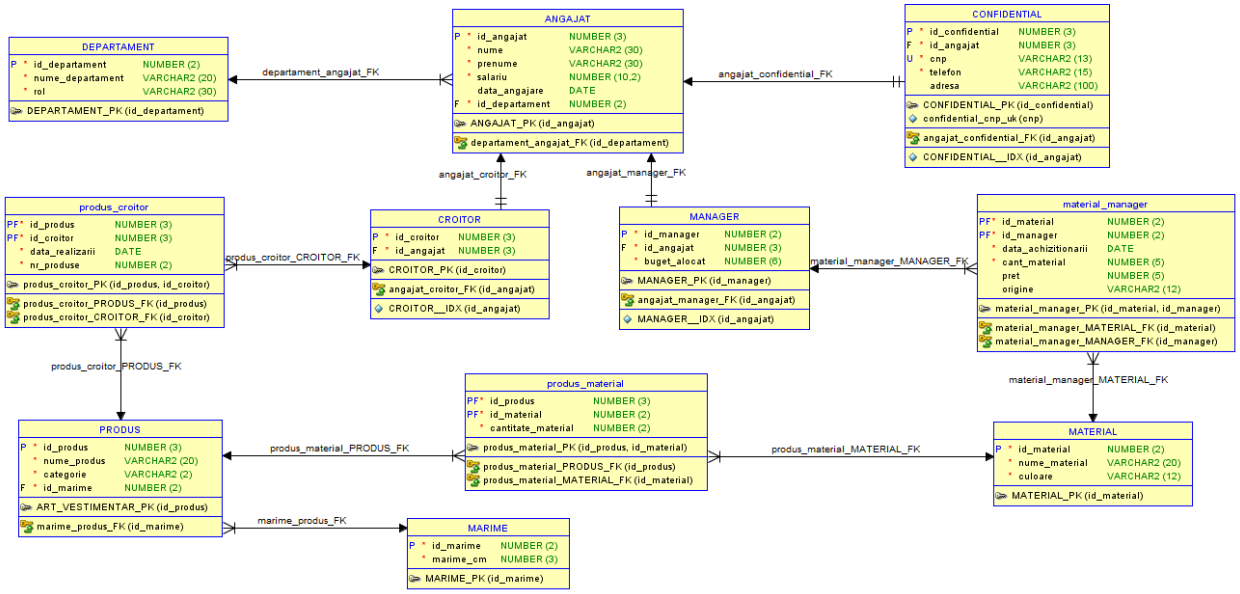

Figura 1.1: Diagrama ER a bazei de date

Capitolul 2. Descrierea detaliata a entitatilor si a relatiilor dintre tabele

Tabelele din aceasta aplicatie sunt**:**

- ✓ ANGAJAT
- ✓ DEPARTAMENT
- ✓ CONFIDENTIAL
- ✓ CROITOR
- ✓ PRODUS_CROITOR
- ✓ PRODUS
- ✓ MARIME
- ✓ PRODUS_MATERIAL
- ✓ MATERIAL
- ✓ MATERIAL_MANAGER
- ✓ MANAGER

In proiectarea acestei baze de date s-au identificat tipurile de relatii 1:1, 1:n si m:n.

Intre tabelele department si angajat se intalneste o relatie de tip 1:n deoarece dintr-un departament pot face parte mai multi angajati . Reciproca insa nu este valabila, deoarece un angajat poate sa faca parte dintr-un singur departament,adica are o singura responsabilitate in atelier.

Inre tabelele angajat si confidential se stabileste o relatie 1:1 . Un angajat are un singur set de date confidentiale,iar datele ii corespund unui singur angajat.

De asemenea intre tabela angajat si tabelele croitor si manager se stabileste o relatie de tip 1:1.

Intre croitor si produs se stabileste o legatura de tip m:n deoarece un croitor poate realiza mai multe produse ,iar un produs poate fi confectionat de catre mai multi croitori.Aceasta legatura va fi sparta in doua. Se vor forma doua relatii 1:n si legatura intre cele doua tabele se va mentine prin tabela produs_croitor. Legatura se face prin cele doua chei primare id_produs si id_croitor . Tabela nou creata mai contine o data calendaristica ,data_realizarii unui produs

,dar si un atribut numeric nr_produse cu ajutorul carora vom raspunde la intrebarea: Cate produse a creat un croitor la o data ? .

De asemenea ,aceeasi relatie de m:n se stabileste si intre tabela manager si tabela material . Un manager poate achizitiona mai multe tipuri de material,iar un tip de material poate fi procurat de catre mai multi manageri.Aceasta relatie se va sparge in doua ,rezultand doua relatii 1:n si legatura intre cele doua tabele se va realiza cu ajutorul unei alte tabele “material_manager” care va contine cheia primara a fiecarei din cele doua tabele.Altfel spus legatura se face prin doua campuri id_manager si id_material reunite intr-o tabela comuna. Aceasta tabela de legatura mai contine data_achizitionarii unui material ,atributul cant_material ce reprezinta numarul total de metri material achizitionat .Mai avem si un pret ,ce reprezinta valoarea unui metro de material de un anumit tip si in acelasi timp un atribut origine ce reprezinta tara din care a fost procurat materialul de catre manager.

Intre marime si produs se stabileste o relatie 1:n . Aceeasi marime o putem intalni la mai multe produse ,insa un produs poate avea o singura marime.

In final mai avem o legatura de tip m:n intre produs si material ,stabilita astfel deoarece un produs poate fi confectionat din mai multe material,iar un material se poate gasi in component mai multor produse. La fel se va sparge relatia in doua de tip 1:n,iar legatura dintre cele doua tabele va fi pastrata prin noua tabela “produs_material” ce contine cele doua chei primare id_produs si id_material ,dar si o entitate cantitate_material prin care vom putea raspunde la intrebarea: Ce cantitate de material este necesara la realizarea unui anumit produs?

Capitolul 3. Descrierea constrangerilor folosite si de ce au fost acestea necesare

Folosirea constrangerilor de integritate referentiala:

- ✓ **primary key/foreign key**

Fiecare tabel in parte prezinta un camp de tip index cu auto-incrementare care respecta o constrangere de tip Primary Key. Deci este asigurata existent unei instante unic identificabila in tabele. Constrangerile de tip Foreign key sunt folosite pentru a crea relatii intre cheile primare ale altor tabele.

Folosirea constrangerilor de integritate de tip:

- ✓ **Check**

Aest tip de constrangere a fost folosit pentru a nu permite introducerea atributului telefon in mod gresit in tabela confidential. Numarul de telefon trebuie sa contina 10 numere ,primul numar sa fie egal cu 0 ,iar al doilea sa fie egal cu 2,3 sau 7.

In acelasi timp se foloseste aceasta constrangere pentru a da atributului buget_alocat din tabela manager valori in intervalul 10000 – 500000 lei.

Atributul marime_cm din tabelul marime ia doar valorile din lista mentionata mai sus de marimi exprimate in cm.

Atributul nume_material din tabela material contine doar denumirile din lista : ('bumbac', 'bumbac_m', 'bumbac_s', 'in', 'stofa').

Constrangerea atributului nume_produs din tabela produs are ca efect folosirea doar a valorilor din lista : ('camasa', 'fusta', 'hanorac', 'palton', 'pantalon', 'pijama', 'rochita', 'sacou', 'salopeta', 'tricou').

De asemenea constrangerea atributului categorie din tabela produs are ca efect folosirea doar a valorilor din lista: ('B', 'BB', 'BG', 'G').

Deci atelierul isi propune sa creeze doar aceste produse si doar din aceste materiale.

- ✓ **Unique**

Inafara de acele atribute care sunt Primary Key si respecta implicit constrangerea de tip Unique, in proiect exista un atribut care este necesar sa fie unic: pentru tabela confidential, atributul cnp.

- ✓ **not null**

Acest tip de constrangere a fost folosit pentru a specifica campurile esentiale pentru o entitate. Pentru ca majoritatea atributelor din toate entitatile respecta acest tip de constrangere, le voi specifica doar pe cele care nu sunt esentiale :

- Tabelele ANGAJAT(data_angajare) si CONFIDENTIAL(adresa) contin campuri care nu sunt esentiale.

Capitolul 4. Comenzi disponibile în aplicație

Aplicația permite utilizatorului realizarea operațiilor de căutare, adăugare, update, și ștergere a datelor din tabele.

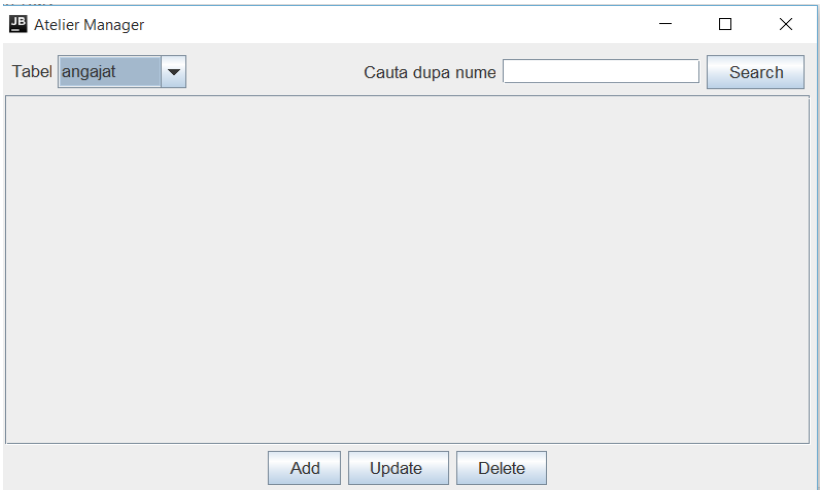

Figura 4.1: Fereastra principala

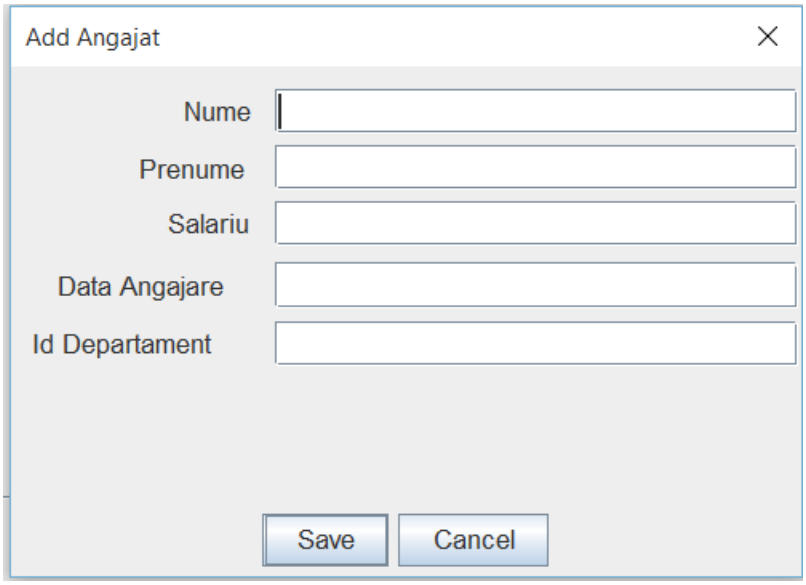

Figura 4.2: Fereastra de adaugare date

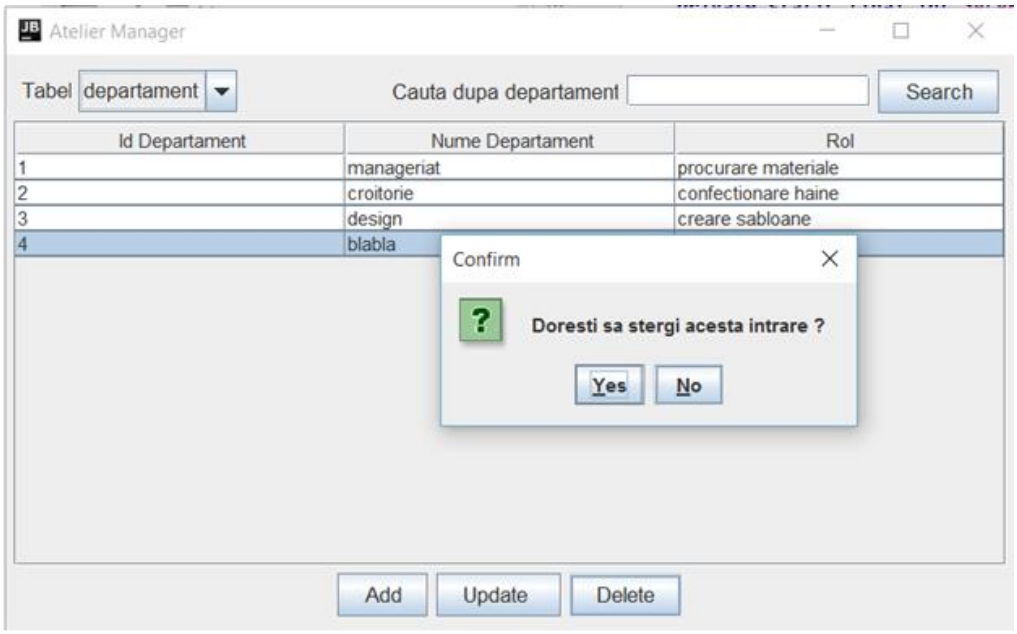

Figura 4.3: Fereastra de stergere

Capitolul 5. Conectarea la baza de date

Aplicația a fost scrisă în limbajul Java, așa încât conectarea la baza de date Oracle se face utilizând drive-ul oficial JDBC pentru Oracle .

```
31      public AngajatDAO(AtelierManagerApp mainFrame) {
32          Properties props = new Properties();
33          try {
34              ClassLoader classLoader = getClass().getClassLoader();
35              props.load(classLoader.getResourceAsStream( name: "atelier.properties"));
36          } catch (IOException e1) {
37              e1.printStackTrace();
38          }
39          String dburl = props.getProperty("dburl");
40          props.setProperty("useSSL", "false");
41          try {
42              Class.forName("oracle.jdbc.driver.OracleDriver");
43              try {
44                  myConn = DriverManager.getConnection(dburl, props);
45                  System.out.println("Conected to Oracle");
46              } catch (SQLException ex) {
47                  ex.printStackTrace();
48              }
49              this.mainFrame = mainFrame;
50
51          } catch (ClassNotFoundException e) {
52              System.out.println(e);
53          }
54
55      }
```

Figura 5.1

Capitolul 6. Exemple instrucțiuni SQL folosite în aplicație

```
144 @   public void addAngajatToDatabase(Angajat theAngajat) throws SQLException {
145         PreparedStatement myStmt=null;
146         try {
147             myStmt=myConn.prepareStatement( s: "insert into angajat"+"(id_angajat,nume, prenume, salariu, data_angajare, id_departament)"+"values(?,?,?,?,?,?)");
148             myStmt.setInt( i: 1,  i1: getLastId()+1);
149             myStmt.setString( i: 2, theAngajat.getNume());
150             myStmt.setString( i: 3, theAngajat.getPrenume());
151             myStmt.setFloat( i: 4, theAngajat.getSalariu());
152             myStmt.setDate( i: 5, theAngajat.getData_angajare());
153             myStmt.setInt( i: 6, theAngajat.getId_departament());
154             myStmt.executeUpdate();
155             sqlError = false;
156         }
157         catch (SQLException e) {
158 
159             String message = e.getMessage();
160             if (message.length() > 65) {
161                 message = message.substring(0, 64);
162             }
163             JOptionPane.showMessageDialog(mainFrame,  message: "Eroare salvare date confidentiale:\n" + message,  title: "Error",
164                     JOptionPane.ERROR_MESSAGE);
165             sqlError = true;
166         }
167         finally {
168             close(myConn, myStmt,  myRs: null);
169         }
170     }
171 
```

Figura 6.1

```
75      public List<Departament> SearchDepartament(String nume) throws SQLException
76      {
77          List<Departament>listDepartament=new ArrayList<~>();
78          PreparedStatement myStmt=null;
79          ResultSet myRs=null;
80          try {
81              nume += "%";
82              myStmt=myConn.prepareStatement( s: "select* from departament where nume_departament like ?");
83              myStmt.setString( i: 1, nume);
84              myRs=myStmt.executeQuery();
85              while(myRs.next())
86              {
87                  Departament tempDepartament=convertRowToDepartament(myRs);
88                  listDepartament.add(tempDepartament);
89              }
90          } catch (SQLException e) {
91 
92              e.printStackTrace();
93          }
94          finally
95          {
96              close(myConn,myStmt,myRs);
97          }
98          return listDepartament;
99      }
```

Figura 6.2