**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE DEPARTAMENTUL CALCULATOARE**

PROIECT

la disciplina

**INTRODUCERE ÎN BAZE DE DATE**



**Profesor coordonator: Echipa de proiect:**

**Georgiana Bianca Cornea Neacă Radu-Sabin**

**Cîrceie Gavrilă-Nicolae**

**Grupa: 30222, semestrul I**

**An academic 2021-2022**

**Cuprins**

1. **Introducere**
   1. Introducere, argumente, scop și obiective specifice
2. **Suportul teoretic**
3. **Analiza cerințelor utilizatorilor (Specificațiile de proiect)**
   1. Ipotezele specifice domeniului ales pentru proiect (cerinte, constrangeri)
   2. Organizare structurată (tabelar) a cerințelor utilizator
4. **Modelul de date și descrierea acestuia**
   1. Entitati și atributele lor (descriere detaliata **– implementarea fizică**)
   2. Diagrama EER/UML pentru modelul de date complet
5. **Detalii de implementare MySQL**
   1. DDL
   2. DML
   3. SQL programatic
6. **Detalii de implementare Java**
   1. Structura de clase în Java
7. **Utilizarea platformei**
8. **Concluzii. Limitări și dezvoltări ulterioare**

**1.Introducere**

Am realizat o interfață grafică cu tema „Magazin Virtual” care permite oricărui utilizator să își creeze un profil utilizator, explorarea produselor existente, cautarea unui produs, adăugarea lui în cos. Produsele adăugate vor fi dispuse pe categorii.

Pentru dezvoltarea proiectului au fost folosite:

**MySQL Workbench 8.0CE** - pentru crearea bazei de date, popularea inițială, dezvoltarea de vederi, proceduri și triggere și pentru crearea diagramei UML a tabelelor

**IntelliJ** – medii de dezvoltare Java

**JDBC**– pentru stabilirea conexiuni aplicației cu baza de date

**2. Suportul teoretic**

**Limbajul MySql**

MySQL este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeza MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open- source la ora actuală, fiind o componentă cheie a stivei LAMP (Linux, Apache, MySQL, PHP).

Deși este folosit foarte des împreună cu limbajul de programare PHP, cu MySQL se pot construi aplicații în orice limbaj major. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum ar fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc., fiecare dintre acestea folosind un tip specific API. O interfață de tip ODBC denumită MyODBC permite altor limbaje de programare ce folosesc această interfață, să interacționeze cu bazele de date MySQL cum ar fi ASP sau Visual Basic. În sprijinul acestor limbaje de programare, unele companii produc componente de tip COM/COM+ sau .NET (pentru Windows) prin intermediul cărora respectivele limbaje să poată folosi acest SGBD mult mai ușor decât prin intermediul sistemului ODBC. Aceste componente pot fi gratuite (ca de exemplu MyVBQL) sau comerciale.

Licența GNU GPL nu permite încorporarea MySQL în softuri comerciale; cei care doresc să facă acest lucru pot achiziționa, contra cost, o licență comercială de la compania producătoare, MySQL AB.

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 ca 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”.

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. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, phpMyAdmin.

MySQL poate fi rulat pe multe dintre platformele software existente: AIX, FreeBSD, GNU/Linux, Mac OS X, NetBSD, Solaris, SunOS, Windows 9x/NT/2000/XP/Vista.[2]

**Limbajul Java**

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către James Gosling la Sun Microsystems (acum filială Oracle) la începutul anilor ʼ90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile, spre exemplu telefon, agenda electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intranet-urilor.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java (engleză Java Virtual Machine, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (engleză byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF. În 2006, Sun a anunțat că face disponibilă varianta sa de JVM ca open-source.

**3.Analiza cerințelor utilizatorilor**

*3.1. Ipotezele specifice domeniului ales pentru proiect*

* Exista 2 tipuri utilizatori: client și admin
* Un client este unic identificat prin idCont. Pentru acesta mai reținem nume, prenume, adresa, data nașterii.
* Există mai multe tipuri de produse definite prin IdProdus, Denumire și categoria de care aparțin.
* Există un serviciu de comandă, cu ajutorul căruia faci comanda. Comanda este identificată prin NrComandă.

*3.2. Organizare structurată tabelar a cerințelor utilizatorilor*

Baza de date trebuie să stocheze următoarele informații:

* Produse
* Furnizor
* Recenzia unui client
* Comanda
* Tranzacție
* Categorie

**4. Modelul de date și descrierea acestuia**

*4.1. Entități și atributele lor*

Tabelele:

**Produs** – informatii despre produse. Atribute: IdProdus (PK), Denumire, Garantie, Stoc, ValoareUnitara, Valuta.

**Categorie –** informații despre categoria unui produs. Atribute: Cod (PK), Denumire.

**Furnizor –** informații despre furnizori. Atribute: CodFurnizor (PK), Denumire, Adresa.

**Recenzie –** informații despre recenzia unui produs. Atribute: Id (PK), Persoana, DataRecenzie, Nota, Descriere,

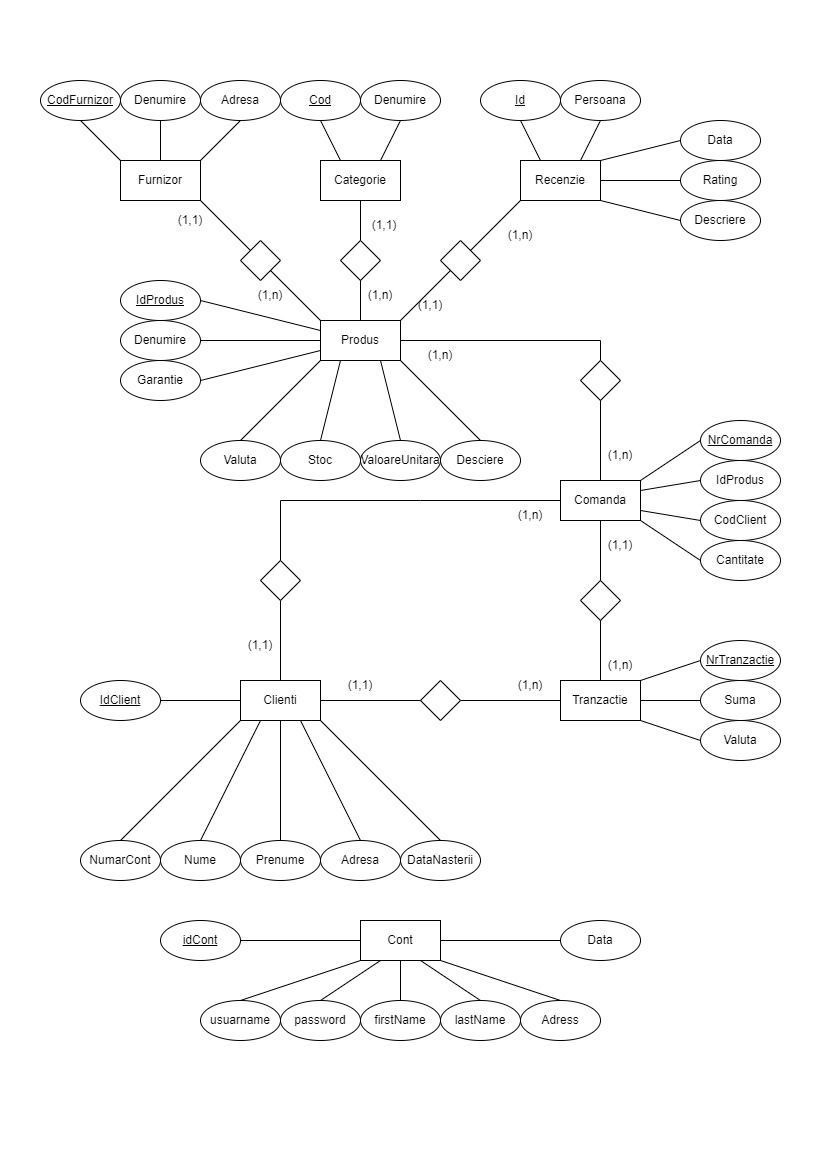
**Clienți –** informații despre client

**Comandă –** informații legate de comandă. Atribute: NrComanda (PK), IdProdus, CodClient, Cantitate, Id\_Comanda.

**Tranzacție –** informații despre tranzacția realizată de către client. Atribute: NrTranzactie (PK), ValoareTotala, Valuta, Id\_comanda, Id\_client.

**Cont –** informații legate de datele de autentificare ale unui client. Atribute: idCont, usernameCont, passwordCont, firstName, lastName, Adress, dataNasterii.

*4.2. Diagrama EER/UML pentru modelul de date complet*

**

**5. Detalii de implementare SQL**

*5.1. DDL (Data Definition Language)*

* *Crearea bazei de date*

create database magazin\_virtual;

use magazin\_virtual;

* *Crearea unui tabel*

CREATE TABLE IF NOT EXISTS produs (

IdProdus INTEGER NOT NULL PRIMARY KEY,

Denumire VARCHAR(50),

Garantie INTEGER,

Stoc INTEGER,

ValoareUnitara INTEGER,

Valuta VARCHAR(10),

#CodCategorie INTEGER,

#CodFurnizor INTEGER,

Id\_Categorie INTEGER,

FOREIGN KEY (Id\_Categorie)

REFERENCES Categorie (Cod),

Id\_Furnizor INTEGER,

FOREIGN KEY (Id\_Furnizor)

REFERENCES Furnizor (CodFurnizor) );

* *Adaugarea cheilor străine*

Id\_Produs INTEGER,

FOREIGN KEY (Id\_Produs)

REFERENCES produs (IdProdus)

*5.2. DML (Data Manipulation Language)*

* *Inserarea datelor în tabele*

INSERT INTO produs(IdProdus, Denumire, Garantie, Stoc, ValoareUnitara, Valuta) VALUES

(1,'Fujitsu Siemens Amilo Pro',1,10,2000,'RON'),

(2,'Indesit WLI1000 ',3,5,900,'RON'),

(3,'Gorenje RC400',3,4,1500,'RON'),

(4,'Smartphone xyz',2,9,2000,'RON'),

(5,'PC MYRIA Style V71',2,1,5500,'RON');

*5.3. SQL programatic*

* **Proceduri stocate**
* Procedura pentru scăderea produselor din stoc

delimiter //

create procedure client\_produs (NumarProduse integer, id\_produs integer)

begin

update produs P

set Stoc = Stoc - NumarProduse where id\_produs = P.IdProdus;

end; //

* Triggere
* Trigger pentru scăderea unui singur produs din stoc

delimiter //

create trigger produs\_comanda after insert on produs\_has\_comanda

for each row begin

update produs P

set P.Stoc = P.Stoc - 1 where

P.IdProdus = ANY (select id\_produs from produs\_has\_comanda);

end //

* View
* View pentru raport cumpărături

create view raport\_cumparaturi as

select C.Nume, C.Prenume, P.Denumire, K.Cantitate, T.ValoareTotala

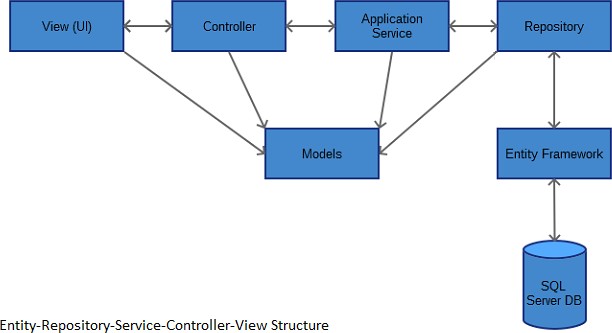
from clienti C, produs P, comanda K, tranzactie T

where ((P.IdProdus = K.NrComanda) &&( T.NrTranzactie = C.IdClient)&&

( C.IdClient = K.NrComanda));

**6. Detalii de implementare Java**

*6.1. Structura de clase în Java*



* **Scurtă prezentare**

Structura ierarhică Entitaty-Repository-Service-View-Controller structurează aplicația noastră pe 5 nivele distincte:

* Enitity – realizează maparea bazei de date;
* Repository – asigură comunicarea cu baza de date;
* Service – se folosește de mai multe clase de Repository pentru a asigura funcționalitatea specifică cerințelor;
* View – îl ajută pe utilizator să interacționeze cu aplicația într-o manieră interactivă;
* Controller - face legătura dintre Service și View și controlează funcționalitatea;
* Serviciu de Login

@Override

public void actionPerformed(ActionEvent e) {

//click the reset button

if(e.getSource()==newAccountButton) {

frame.dispose();

AccountSettings accountSettings = new AccountSettings();

}

//login data

if(e.getSource()==loginButton) {

String userID = userIDField.getText();

String password = String.valueOf(userPasswordField.getPassword());

if(logininfo.containsKey(userID)) {

if(logininfo.get(userID).equals(password)) {

messageLabel.setForeground(Color.green);

messageLabel.setText("Login successful");

frame.dispose();

SearchProduct searchProduct = new SearchProduct();

}

else {

messageLabel.setForeground(Color.red);

messageLabel.setText("Wrong password");

userIDField.setText("");

userPasswordField.setText("");

}

}

else {

messageLabel.setForeground(Color.red);

messageLabel.setText("username not found");

userIDField.setText("");

userPasswordField.setText("");

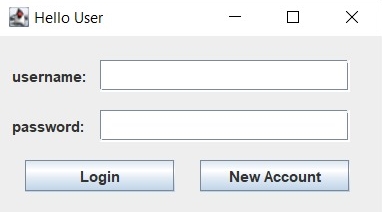
}

}

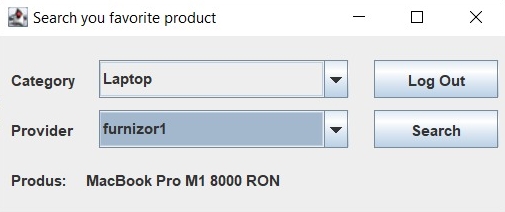
}

**7. Utilizarea aplicației**

* Pagina de logare



* Pagina de căutare

****

**8.Concluzii. Limitări și dezvoltări ulterioare**

Proiectul nostru oferă gestionarea în mod eficient a sistemului informatic al unei platforme de magazin virtual, în cadrul careia sunt oferite diferite funcționalități pentru utilizatorii acesteia, care pot fi clienți și admin**.** Aplicația noastră oferă o interfață minimalistă, este ușor să navighezi prin ea, clienții noștri pot să vadă furnizorii produselor, categoria și tipul de produs.

Niciun utilizator nu poate accesa baza de date dacă nu are creat în prealabil un cont.

Dezvoltările ulterioare pot apărea în cazul interfeţei, unde se pot dezvolta mai multe opţiuni la panoul de control. O altă posibilitate de dezvoltare constă în creearea unui panou de control pentru administrator.

Interfaţa poate fi îmbunătăţită la nivelul design-ului, de exemplu prin adăugarea imaginilor sau prin adăugarea culorilor etc.