**PROIECT**

***LANȚ DE LIBRĂRII***

**Pop Cristian**

**Grupa 30233**

**2020**

**CUPRINS**

1. **Specificarea cerințelor (enunț problemă)**
2. **Instrumente utilizate**
3. **Etapa de analiză**
   1. **Diagrama cazurilor de utilizare**
   2. **Diagrame de activități**
4. **Etapa de proiectare**
   1. **Diagrame de clase**
   2. **Diagrame de secvență**
5. **Prezentarea aplicației**
6. **Specificarea cerintelor**

Cerinta 1

Dezvoltați (analiză, proiectare, implementare) o aplicație desktop care poate fi utilizată într-un lanț de librării. Aplicația va avea 2 tipuri de utilizatori: angajat al lanțului de librării și administrator. Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

 Vizualizarea tuturor cărților dintr-o librărie selectată;

 Filtrarea listei cărților după următoarele criterii: domeniu, disponibilitate, editura, autor, preț;  Căutarea unei cărți după titlu;

 Vizualizarea unor statistici legate de cărțile din acest lanț de librării: procente după domeniu, disponibilitate, editura, autor utilizând grafice (structură radială, structură inelară, de tip coloană, etc.);

 Vânzarea unei cărți (decrementarea cu 1 a numărului de exemplare disponibile din aceea carte);  Operații CRUD în ceea ce privește persistența cărților;

 Salvare rapoarte / liste cu cărțile în mai multe formate: csv, xml, json. Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

 Operații CRUD pentru informațiile legate de utilizatorii de tip angajat. Constrângeri ale aplicației:

 Datele utilizate în aplicație vor fi socate într-un fișier binar.

 Utilizați șabloanele de proiectare creaționale Factory Method și Builder.

Cerinta 2

Obiectiv: Obiectivul acestei teme este familiarizarea cu șablonul architectural Client/Server.

Cerințe: Transformați aplicația desktop implementată la tema 1 într-o aplicație client-server. Mare parte din modelul logic (inclusiv persistența) va face parte din server, mai puțin partea care poate fi determinată din datele primite de la server (generare rapoarte, statistică, etc.).

 În faza de analiză realizați diagrama cazurilor de utilizare.

 În faza de proiectare realizați diagrama de clase corespunzătoare aplicației server și diagrama de clase corespunzătoare aplicației client.

 În faza de implementare scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

 Finalizarea temei va consta în predarea unui director ce va cuprinde:

 Un fișier word care cuprinde numele studentului, grupa, enunțul problemei și instrumentele utilizate.

 Un fișier cu diagramele UML realizate;

 Directorul cu aplicația implementată.

Constrângeri ale aplicației:

 Datele utilizate în aplicație vor fi socate într-o bază de date (MySQL, SQL Server, Oracle, etc.).  Se cere utilizarea arhitecturii client/server, NU web client/server.

Cerinta 3

Obiectiv: Obiectivul acestei teme este familiarizarea cu șablonul de proiectare comportamental Observer.

Cerințe: Optimizați aplicația client/server implementată la tema 2 astfel încât să utilizați șablonul de proiectare comportamental Observer în dezvoltarea aplicației client. În plus, interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională (implicit limba română).

 În faza de analiză realizați diagrama cazurilor de utilizare și diagramele de activități corespunzătoare tuturor cazurilor de utilizare.  În faza de proiectare realizați:

 diagrama de clase corespunzătoare aplicației server;

 diagrama de clase corespunzătoare aplicației client;

 diagrame de secvență corespunzătoare tuturor cazurilor de utilizare.

 În faza de implementare scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

 Finalizarea temei va consta în predarea unui director ce va cuprinde:

 Un fișier word care cuprinde numele studentului, grupa, enunțul problemei, instrumentele utilizate și detalii despre modul în care a fost utilizat șablonul de proiectare Observer.

 Un fișier cu diagramele UML realizate;

 Directorul cu aplicația implementată.

Observație: Pentru persistență se va utiliza o bază de date relațională (MySQL, SQL Server, Oracle, etc.). Se va utiliza baza de date doar în aplicațiile unde este necesară persistența.

1. **Instrumente utilizate**

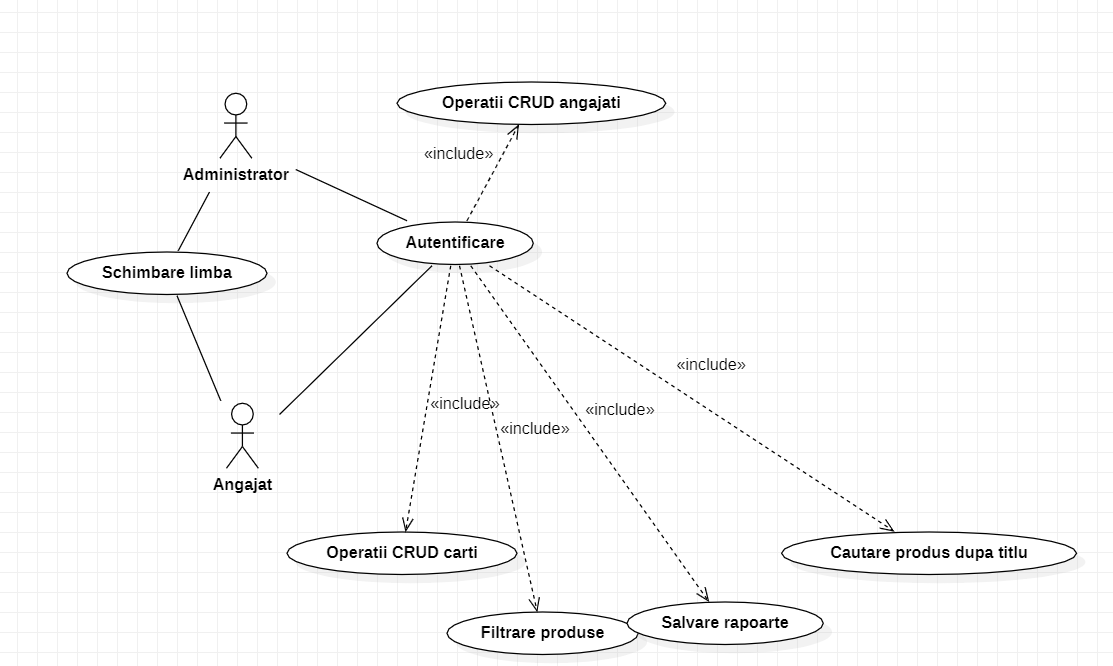
Proiectul a fost realizat in limbajul de programare Java,iar IDE-ul folosit pentru a scrie codul este IntelliJ IDEA Community Edition 2018.Aplicația a fost realizată folosind JavaSwing deoarece sunt mai familiar cu acesta interfață grafică si am mai lucrat cu JavaSwing.

Proiectul conține o bază de date care a fost realizată folosing MySQL,iar IDE-ul utilizat este MySQL Workbench 8.0 CE.Baza de date este legată de aplicatia Java folosind un jar numit mysql-connector-java-8.0.19.

De asemenea pentru realizarea raportului in format JSON am folosit in jar numit json-simple-1.1

1. **Etapa de analiză**

**3.1Diagrama cazurilor de utilizare**

****

Figură -Diagrama cazurilor de utilizare

După cum este specificat și în cerință aplicația conține doi actori.Primul actor este administratorul,iar al doilea este angajatul.Ambii actori pot efectua operația de autentificare,fiind necesară pentru ca aplicația sa știe care tip de actor o folosește.De asemenea ambii actori mai pot efectua oprația de schimbare de limbă.Dupa autentificare ei vor putea efectua diferite operații specifice fiecăruia.

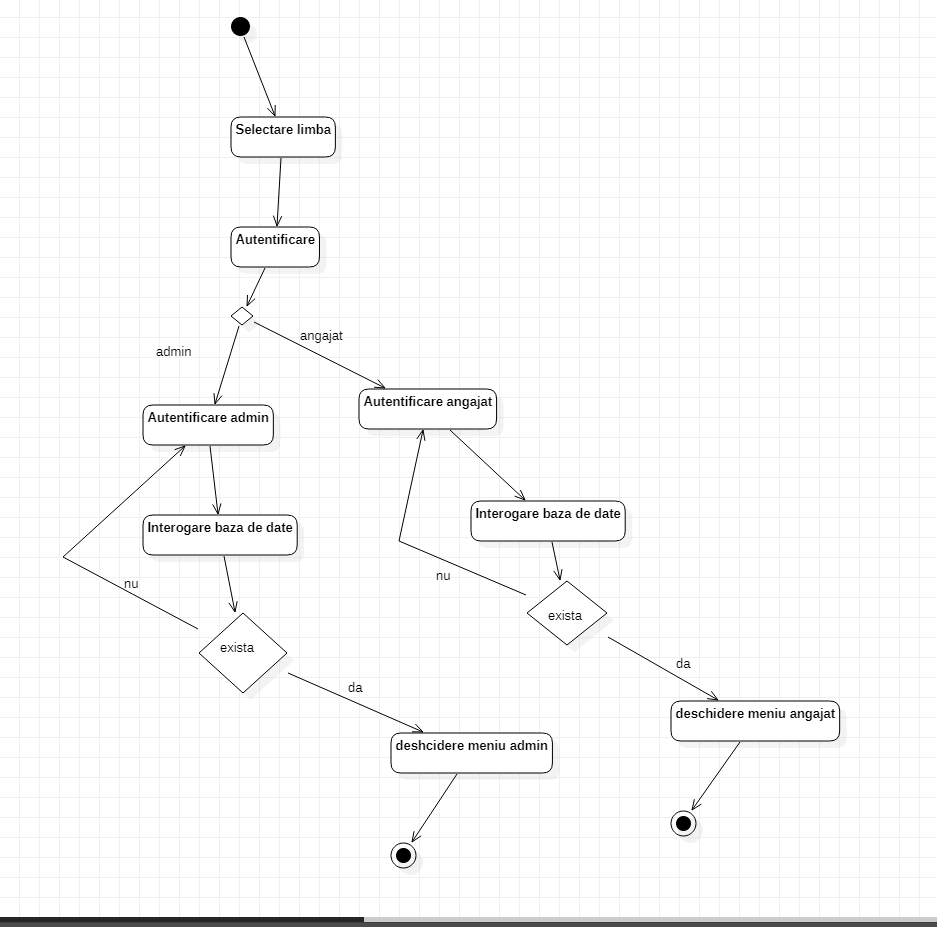
Administratorul va putea efectua operatii CRUD,adica vizualizare,adăugare,ștergere si actualizarea clienților din baza de date.

Angajatul va putea efectua operatii CRUD(vizualizare,adăugare,ștergere si actualizare) pentru cărțile din baza de date.El va mai putea de asemenea căuta o anumita carte sau cărți(dacă există mai multe cărți cu acesași titlu),va putea filtra cărțile dupa urmatoarele criterii:filtrare după autor,editura,domeniu,preț,disponibilitate,librărie.Angajatul mai poate face încă o operație si anume salvarea rapoartelor despre cărți in diferite formate.Aceste formate sunt CSV,JSON și XML.

**3.2Diagrame de activități**

Am realizat mai multe diagrame de activități care sa arate operațiile pe care le poate realiza aplicația.

Prima diagramă de activitate prezentata este cea de autentificare fiind si prima operație pe care trebuie administratorul sau angajatul sa o execute pentru a putea folosi aplicația.

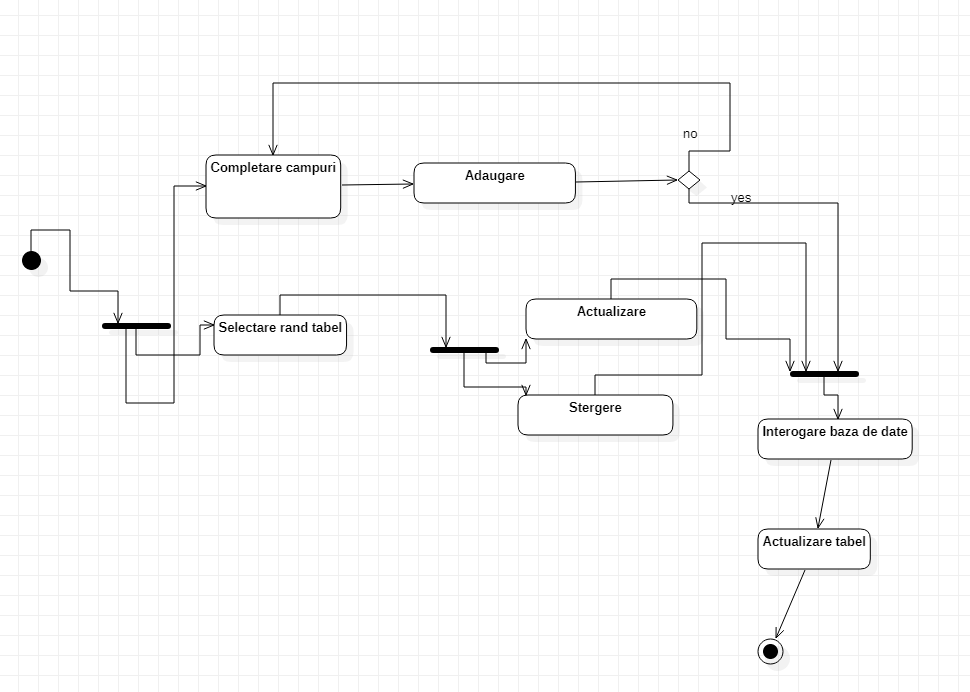
****

Figură 2-Diagrama de activitati autentificare

Înainte de autentificare utilizatorul poate schimba limba.Nu este necesar ca el sa fie autentificat pentru a putea face aceasta operație.

Utilizatorul va alege daca vrea să se autentifice ca si administrator sau ca si angajat.După introducerea datelor adica nume utilizator si parolă se va verifica dacă datele introduse sunt corecte,adica dacă utilizatorul si parola sunt in baza de date a plicației.Dacă nu sunt atunci se va returna un mesaj de eroare,iar utilizatorul va trebui să introducă din nou numele si parola.Daca datele introduse sunt corecte atunci se va deschide o fereastră corespunzătoare tipului de utilizator.

Diagrama de activitati pentru operațiile CRUD.

****

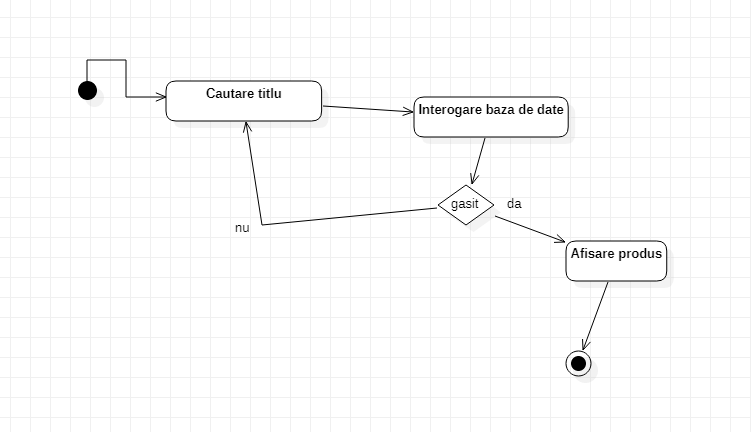
Figură 3-Diagrama activitati operatii CRUD

Operațiile CRUD sunt la fel atat pentru angajat,cât si pentru administrator,singura diferență este că administratorul realizeaza operații CRUD asupra angajaților,iar angajatul realizează operații CRUD asupra cărților.

În cazul in care utilizatorul dorește sa adauge un element nou in tabel,un angajat sau o carte,va completa câmpurile corespunzatoare cu date corecte,iar apoi daca datele introduse sunt bune,se va adauga in tabel si in baza de date obiectul nou de tip angajat sau carte.Daca datele nu sunt corecte va apărea un mesaj de eroare si vor trebui introduse noi date.

Pentru ștergerea si actualizarea datelor trebuie selectat din tabel rândul care conține elementul pe care vrem sa îl ștergem/actualizăm.Dacă vrem sa il ștergem apăsam pe butonul de ștergere,iar tabelul si baza de date vor fi actualizate.Dacă vrem sa actualizăm atunci va trebui sa completăm cu noile date si dupa ce apăsam butonul de actualizare rândul respectiv din tabel va fi actualizat si de asemenea va fi actualizată si baza de date.

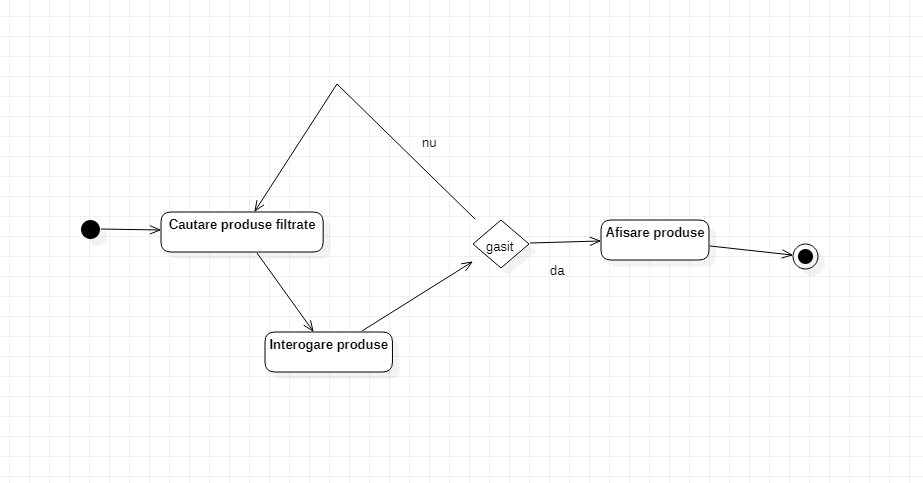
Diagrama de activități pentru căutare titlu.

****

Figură 4-Diagrama activitati cautare titlu

Pentru căutarea unui titlu a unei cărți se va completa câmpul corespunzător apoi se va verifica daca in baza de date există cel putin o carte cu acel titlu.Dacă există atunci in tabel se va afișa doar acea carte sau cărți care au exact acelasi titlu cu cel introdus de angajat.Dacă nu exista nici o carte cu acel titlu nu se va returna nimic si tabelul va fi gol.

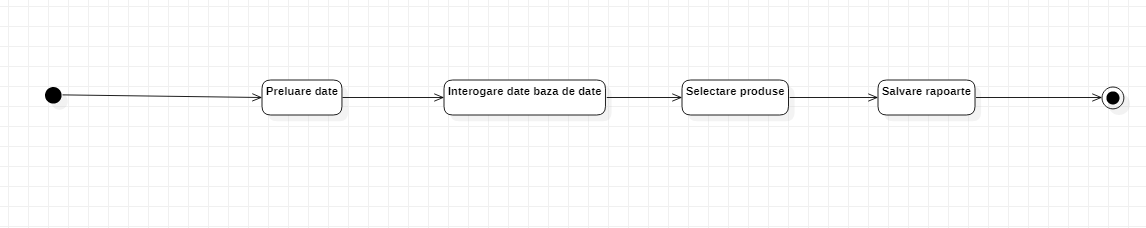
Diagrama de activități pentru filtrarea produselor.

****

Figură 5-Diagrama activitati filtrari

Pentru filtrarea produselor se va completa câmpul corespunzător apoi se va verifica daca in baza de date există cel putin o carte care îndeplinește condițiile impuse de filtrare.Se va afisa cartea sau cărțile care corespund datelor introduse in câmpurile de filtrare.Doar o singura filtrare poate fi aplicată odată,astfel nu este posibilă filtrarea cărților dupa mai multe criterii,doar unul singur pe rând.Daca nu exista niciun produs corespunzător filtrării respective tabelul afișat va fi gol.

Diagrama pentru salvare rapoarte.

****

Figură 6-Diagrama activitati rapoarte

Pentru operația de salvare rapoarte angajatul va prelua datele din tabel si din baza de date,iar apoi apasând butonul de salvare rapoarte va salva obiectele din tabel sub forma de rapoarte in trei formate .csv,.json si .xml.Se va genera câte un fișier de acest tip pentru fiecare raport care poate fi vizualizat.

**4.Etapa de proiectare**

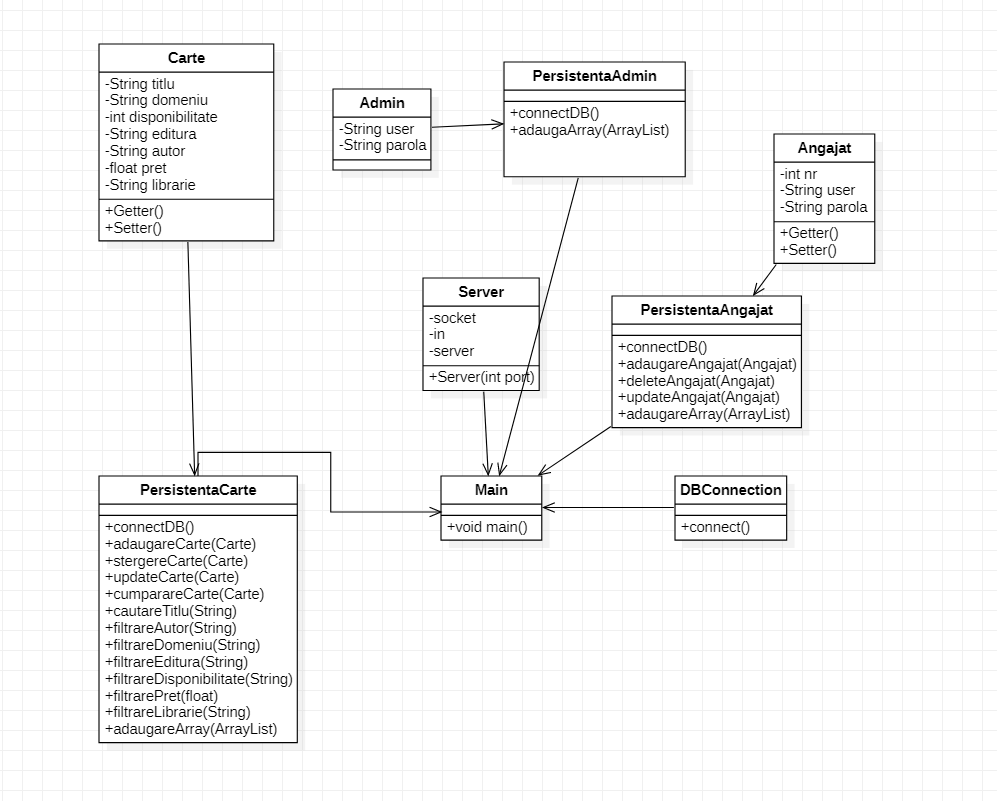
**4.1Diagrama de clase**

Diagrama de clase este împarțită in doua diagrame si anume:

-diagrama de clase pentru server

-diagrama de clase pentru client

Diagrama de clase pentru server.



Figură 7-Diagrama de calse server

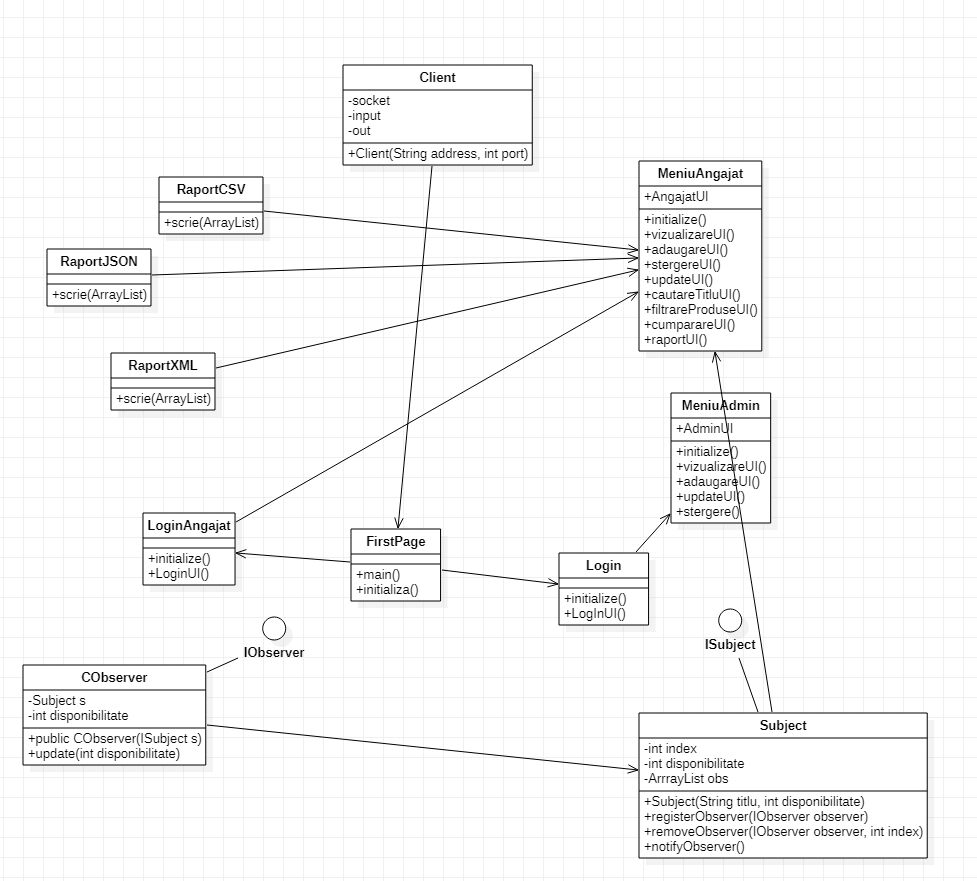
Diagrama de clase a serverului conține în primul rand clasa server care creează serverul.Serverul este creat cu ajutorul unui socket la care se va conecta clientul.

Mai avem si clasele Carte,Admin si Angajat care creeaza obiecte de aceste tipuri.Cu ajutorul acestor clase am creat obiectele necesare aplicației pe care le vom introduce in baza de date.Fiecare din obiectele acestor clase au parametrii specifici cum ar fi pentru Administrator avem user de tip String si parola tot de tip String.Pentru Angajat avem un id de tip int un user de tip String si o parolă tot de tip String.Pentru carte avem urmatorii parametrii:titlu,domeniu,autor,editură si librărie de tip Stirng,disponibilitate de tip integer si preț de tip float.Aceste clase mai conțin gettere si settere corespunzătoare.

Fiecare din clasele de mai sus(Carte,Admin si Angajat) este legată la o alta clasă de persistenta.Avem PersistenaCarte,PersistentaAngajat si PersistentaAdmin.Aceste clase au rolul de a se ocupa cu persistența obiectelor de acel tip.Fiecare din aceste trei clase implementează metoda de connectDB care conectează aplicația la baza de date pentru ca obiectele să poata fi adaugate in baza de date folosing metodele urmatoare.Metodele adaugareCarte,adaugareAdmin,adaugare Angajat adauga un obiect de un anumit tip in baza de date,metoda de ștergere elimina un anumit element din baza de date,metoda de actualizare actualizeaza un element din baza de date,metoda de vizualizare oferă posibilitatea de a vizualiza elementele existente in baza de date.Metoda de adaugareArray din fiecare clasă de persistență este folosită pentru ca obiectele create să fie adaugare într-un ArrayList pentru a fi mai usor de apelat pentru operații.În cazul clasei PersistentaCarte mai avem metode pentru căutare titlu care are ca parametru un String și care returneaza un obiect de tip String care este egal cu acel parametru titlu din metoda.O metoda pentru fiecare tip de filtrare care fiecare returneaza un obiect de tipul ResultSet.Am folosit ResultSet pentru că am considerat ca este mai ușor pentru a afișa obiectele rezultate in urma apelării metodelor.

Mai avem si clasa Main unde este creat obiectul de tip Server si unde este acesta inițializat.

Diagrama de clase pentru client.



Figură 8-Diagrama de clasa client

Diagrama de clase pentru client conține partea de interfață grafica a proiectului.

Diagrama de clase contine clasa Client care creează un obiect de tip Client care se va conecta la server folosind un socket.

Mai avem clasele FirstPage,Login,LoginAngajat,MeniuAdminMeniuAngajat care sunt clase pentru interfața grafica a proiectului.Fiecare din aceste clase inițializeaza un frame diferit.Avem cate un frame pentru pagina de autentificare unde selectăm dacă vrem să ne autentificăm ca și angajat sau administrator,mai avem cate un frame separat pentru fiecare din aceste doua opțiuni unde se introduc datele de autentificare si mai avem încă doua frame-uri pentru fiecare pagina a utilizatorului,adică pentru admin si angajat de unde fiecare din aceștia poate executa oprațiile care le sunt atribuite.

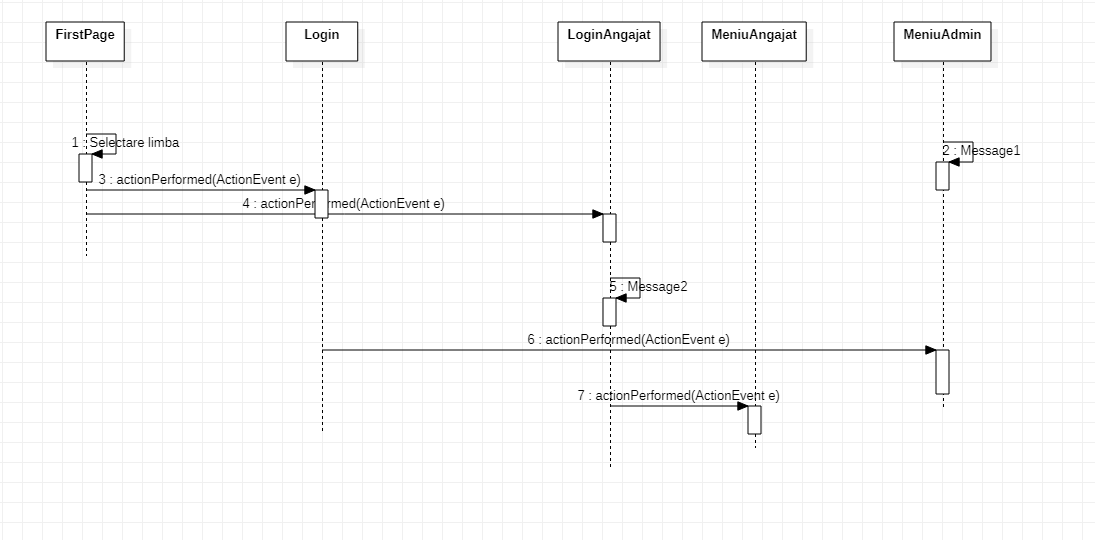
Tot la partea de client mai avem si clasele pentru crearea de rapoarte,câte o clasă pentru fiecare tip de raport.Fiecare din aceste clase are ca si metoda Scrie care primește ca si parametru un ArrayList unde sunt stocate obiectele si apoi acestea sunt scrise in fișierele respective.

La client am implementat Observerul care conține doua clase care implementeaza doua interfețe.Avem interfețele Isubject si Iobserver si clasele Cobserver si Subject care implementează metodele acestor interfețe.

**4.2.Diagrame de secvență**

Am realizat mai multe diagrame de secvență care sa arate toți pașii necesari pentru a realiza operațiile pe care le poate realiza aplicația.

Prima diagramă de secvență prezentata este cea de autentificare fiind si prima operație pe care trebuie administratorul sau angajatul sa o execute pentru a putea folosi aplicația.

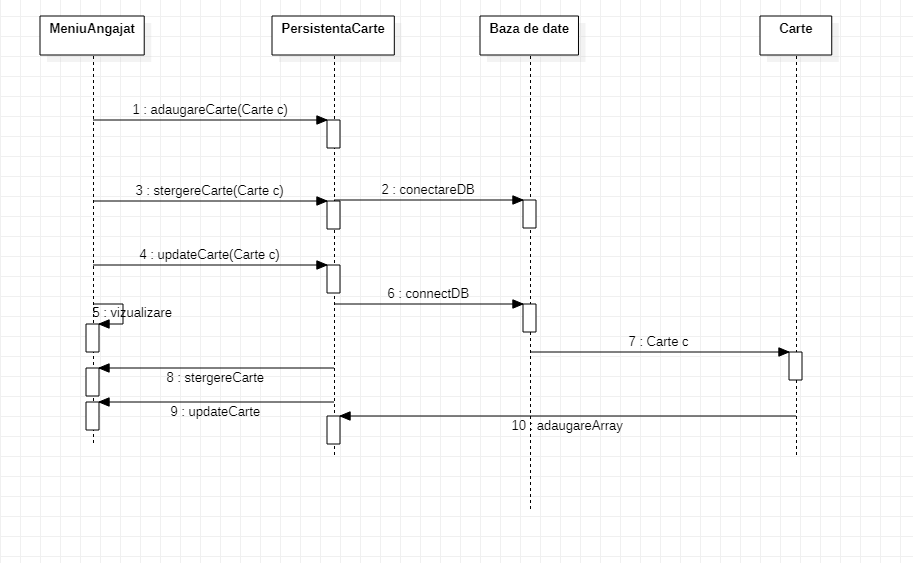


Figură 9-Diagrama secventa autentificare

Se poate schimba limba si fară a fi nevoie de autentificare.

Pentru autentificare trebuie apăsat butonul de autentificare administrator si angajat.In functie de care buton este apăsat se va deschide o fereastră nouă corespunzătoare tipului de utilizator ales folosind metoda actionPerformed.După completarea câmpurilor de utilizator si parolă folosind tot funcția de actionPerformed corespunzătoare butonului de login se va deschide o fereastră in funcție de tipul de utilizator selectat.Dacă datele introduse sunt greșite atunci se va afișa un mesaj de eroare.

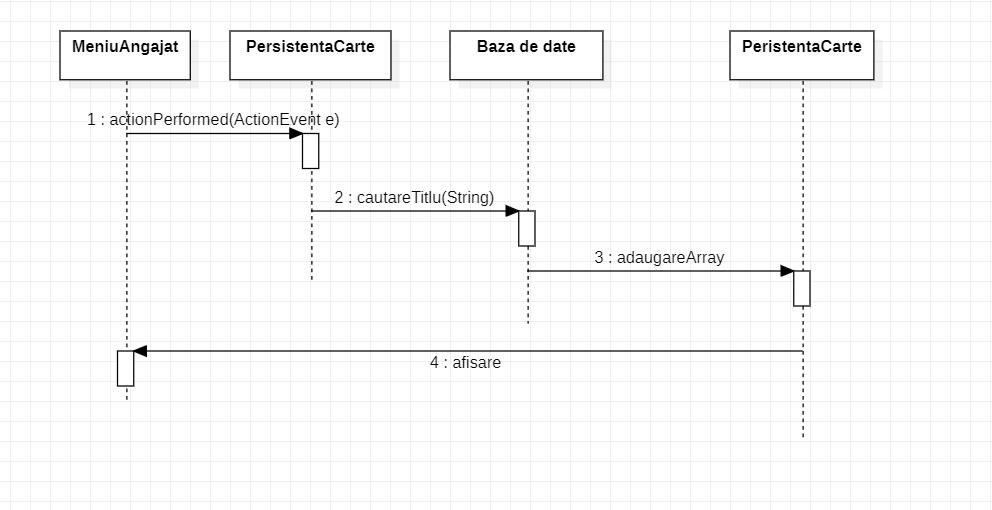
Diagrama de secvență pentu operații CRUD.



Figură 10-Diagrama secventa operatii CRUD

Pentru operațiile CRUD vom utiliza diagrama de secvență pentru operațiile CRUD ale angajatului.Din MeniuAngajat se va selecta operația dorita prin apăsarea butonului corespunzator oprației.Se va apela metoda corespunzătoare din clasa PersistentaCarte.După se va face conectarea la baza de date,iar apoi se va crea un obiect de tip Carte care va fi adaugat in ArrayList-ul de cărți pentru a fi mai usor de efectuat operațiile.Pentru adăugare se va adauga obiectul in ArrayList si se va afisa ArrayListul nou care va conține acel obiect.Pentru operațiile de ștergere si actualizare se va șterge sau actualia elementul dorit si se va afișa ArrayList-ul modificat.

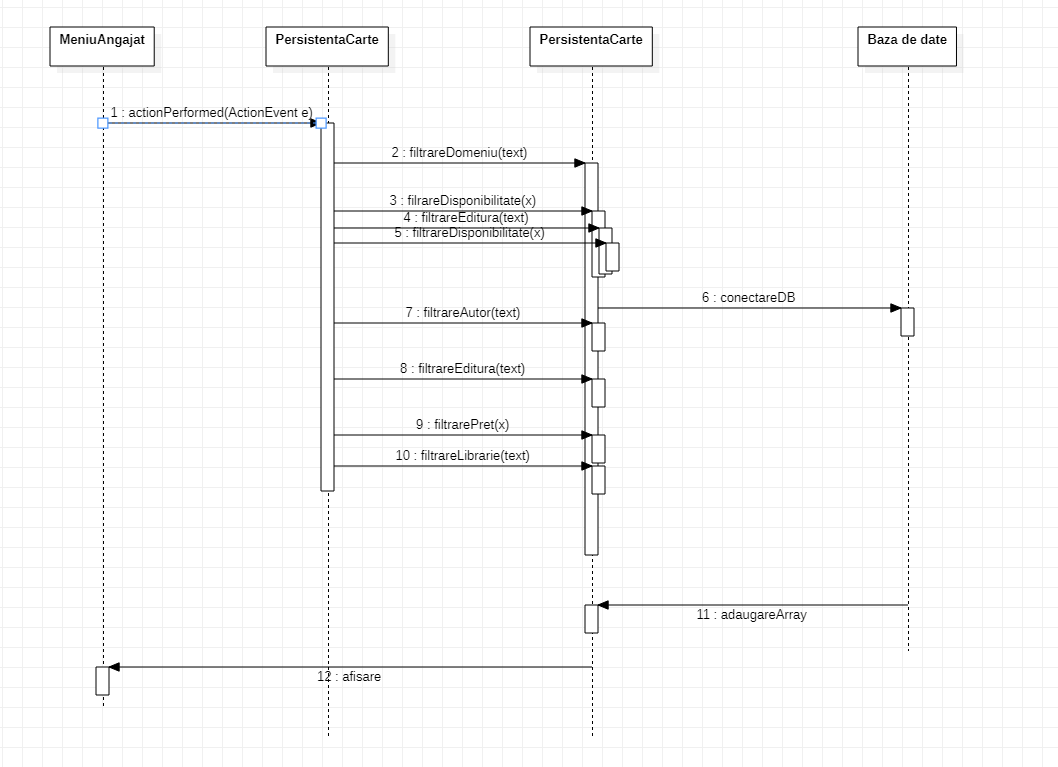
Diagrama de secvență pentru căutare titlu.



Figură 11-Diagrama secventa cautare titlu

În momentul în care apăsam butonul de căutare titlu după ce am introdus in TextField titlul pe care vrem să îl căutăm se va apela functia actionPerformed care va apela metoda de cautareTitlu din PersistentaCarte.Metoda va căuta in baza de date cărțile care au același titlu cu cel introdus de utilizator si va returna acele obiecte ale căror titlu este identic si le va afișa.

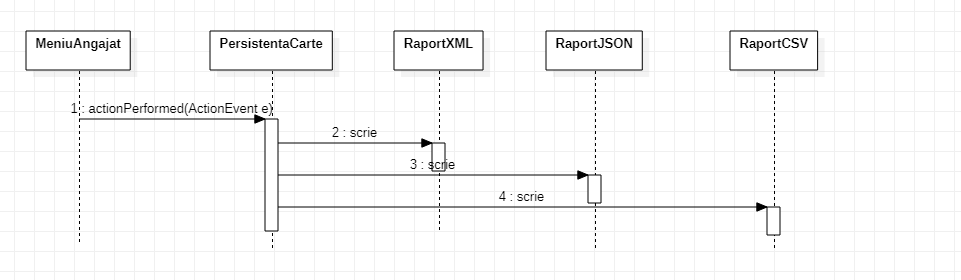
Diagrama de secvență pentru filtrări.



Figură 12-Digrama secventa filtrari

În funcție de filtrare dorită se va apela la in momentul apasarii butonului corespunzător filtrării metoda actionPerformed.Aceasta va crea un obiect de tip PersistentaCarte si va apela metoda corespunzătoare filtrării respective.Această metoda va accesa baza de date si va returna acele obiecte care îndeplinesc condiițiile filtării si anume ceea ce este scris in casuța respectiva sa fie identic cu parametrul corespunzător obiectului.Dacă este cel putin un obiect care respectă aceasta proprietate atunci acel obiect va fi afișat in tabel,dacă nu nu se va afișa nimic.

Diagrama de secvență pentru salvare rapoarte.



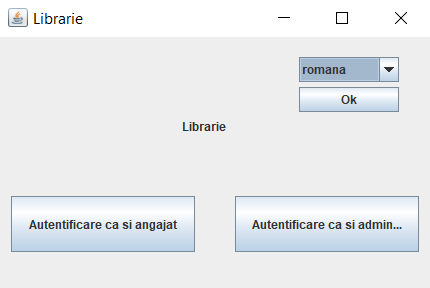
Figură 13-Diagrama secventa rapoarte

Când se apasă butonul de Raport se apelează metoda actionPerformed care va apela din clasa PersistentaCarte metoda de scrie care adauga toate elemntele intr-un ArrayList,iar acel ArrayList va fi scris in fiecare fisier.Pentru fiecare tip de fișier avem o clasa separată cu acceași funcșie de scrie.Astfel datele din tabel sunt salvate in aceste fișiere.

**5.Prezentarea aplicației**

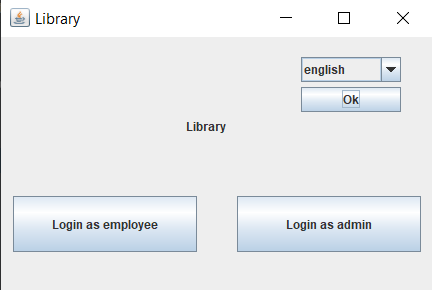
Aplicația implementata este o aplicație pentru managementul unui lanț de librarii.Aplicația este de tip client-server.Este împarțită in server si client.Fiecare din aceste doua proiecte are clasele proprii dupa cum se poate vedea în imaginea de mai jos.

Pentru a putea rula aplicația prima dată se va rula clasa Main din server,iar apoi se va rula FirstPage din client.Se va deschide prima fereastră.



Figură 14-Fereastra principala

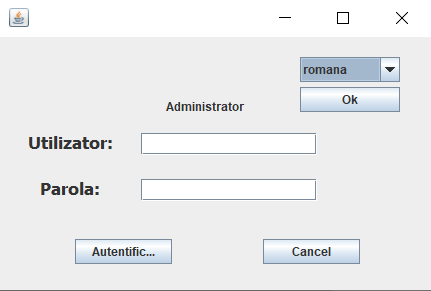
Mai jos puteți vedea aceeasi fereastra,dar cu limba schimbată



Figură 15-Fereastra principala cu limba schimbata in engleza

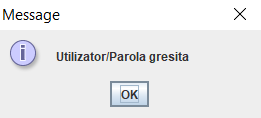
Interfața grafică este realizata folosing JavaSwing.Primul frame conține doua butoane reprezentând tipul de utilizator care vrea sa folosească aplicația.De asemenea mai este un JcomboBox pentru schimbarea limbii.Se selecteaza limba si apoi se apasă butonul Ok.

Daca se apasa butonul de Autentificare ca si administrator se va deschide o fereastră nouă.

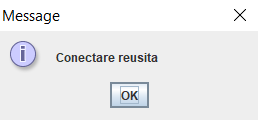


Figură 16-Fereastra de login administrator

Administratorul va trebui să introducă datele corespunzatoare pentru a se putea autentifica.Există doar un singur administrator deci doar o singura variantă de autentificare este corectă.Dacă datele introduse nu sunt corecte se va afisa un mesaj de eroare,iar dacă datele introduse sunt corecte se va afișa un mesaj de succes.

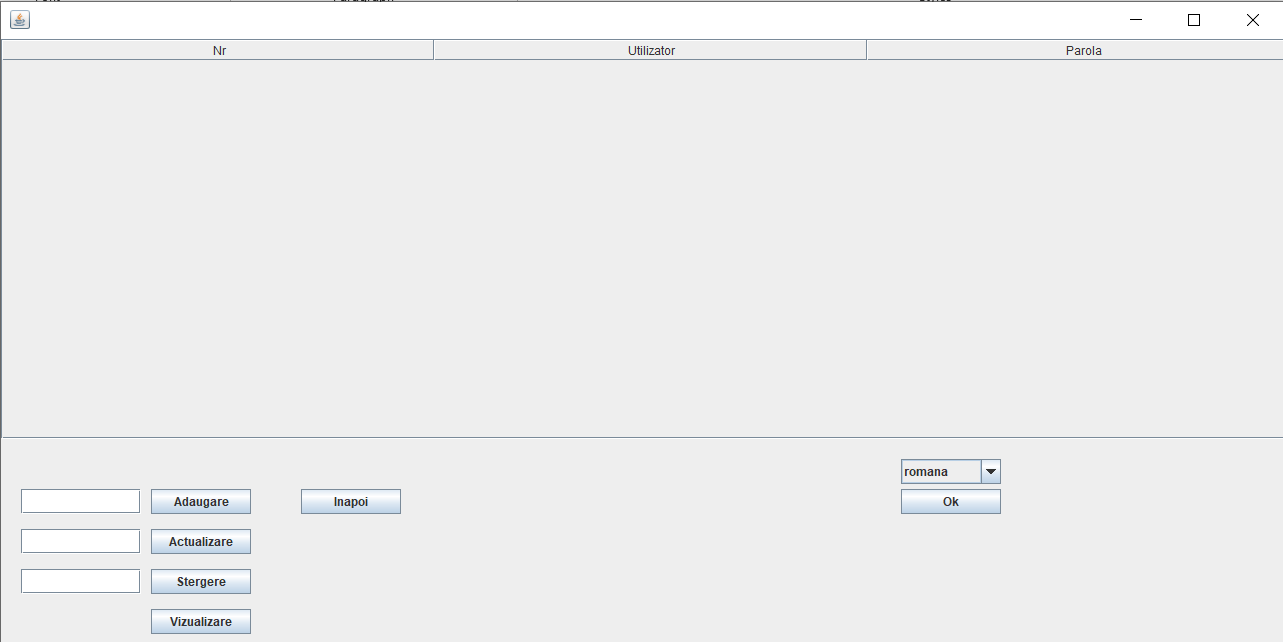


Figură 17-Mesaj de eroare autentificare



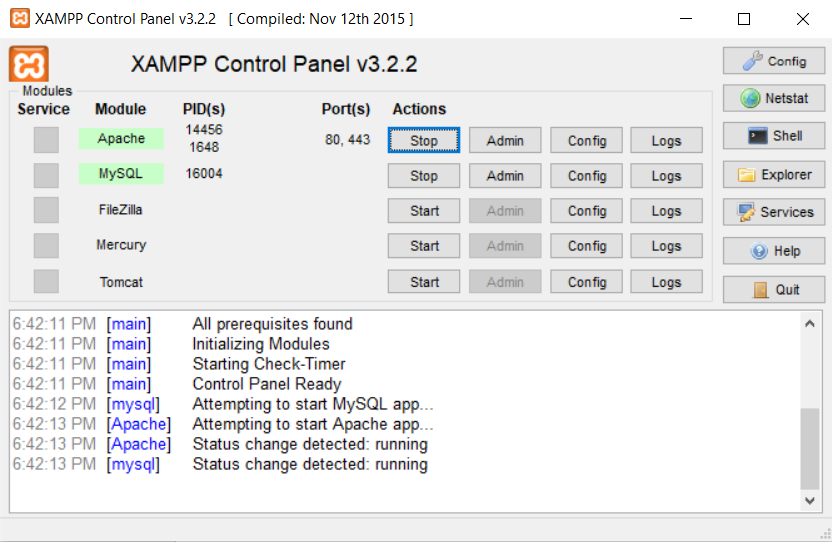
Figură 18-Mesaj autentificare reusita

După ce administratorul se autentifica cu succes se va deschide fereastra lui unde poate efectua opeațiile CRUD asupra angajaților.Mai jos se poate vedea fereastra



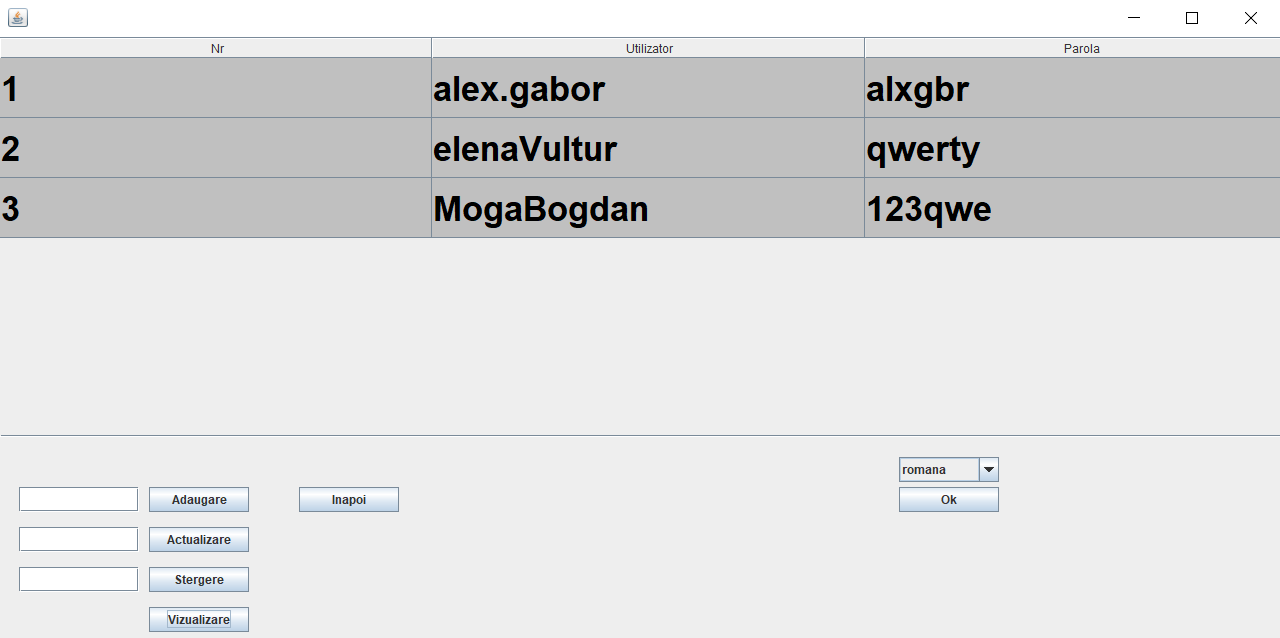
Figură 19-Fereastra meniu administrator

Pentru a putea vizualiza angajații din baza de date prima data va trebui sa pornim XAMPP si sa pornim MySQL si Apache dupa cum se poate vedea in imaginea de mai jos.



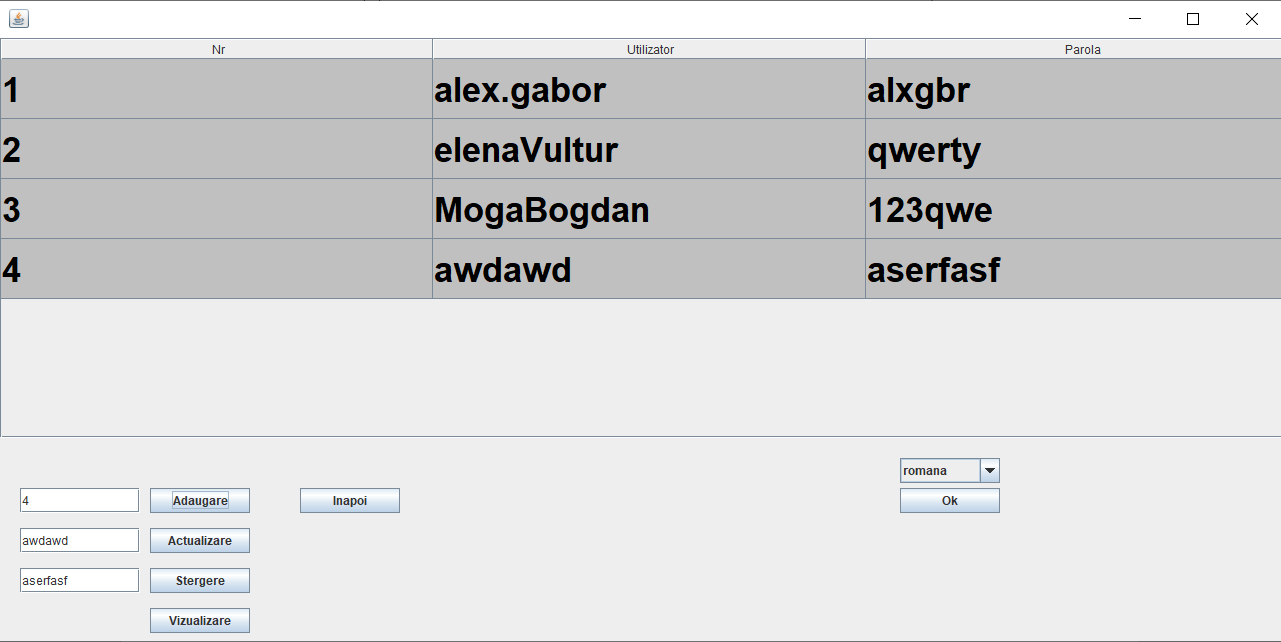
Figură 20-Meniu XAMPP

Dacă XAMPP este pornit atunci se pot vizualiza elementele din baza de date si anume angajații.



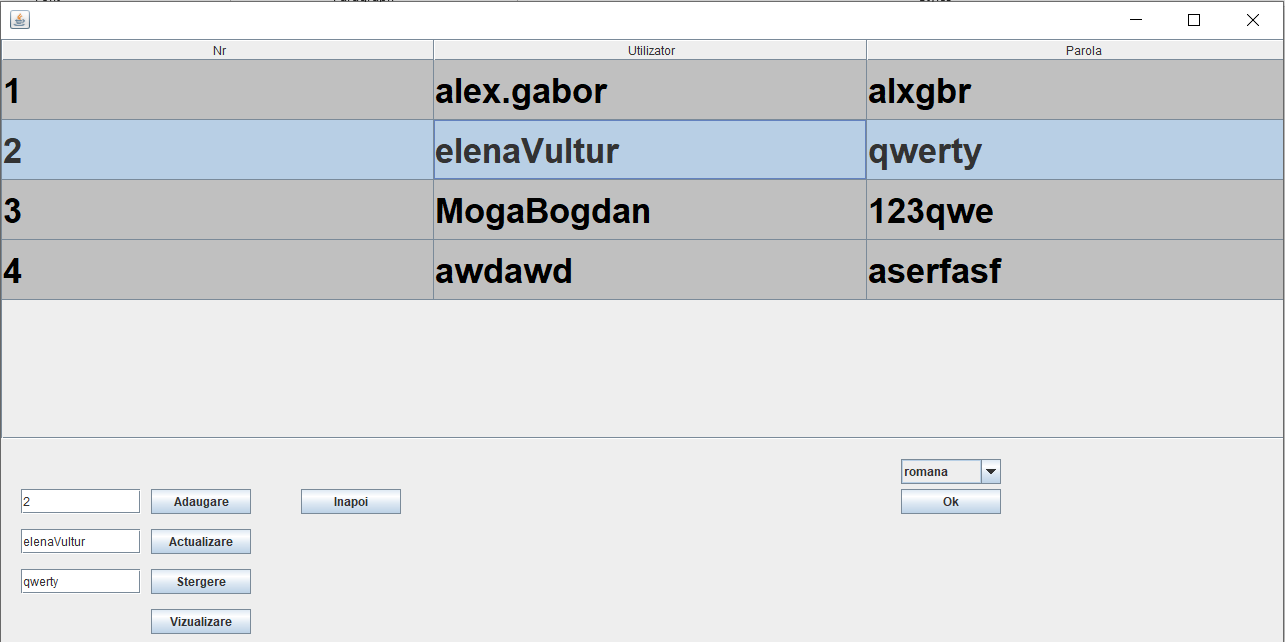
Figură 21-Meniu administrator cu tabel angajati

Dacă dorim sa adaugam un angajat nou vom completa câmpurile cu datele corespunzatoare si anume id user si parolă.

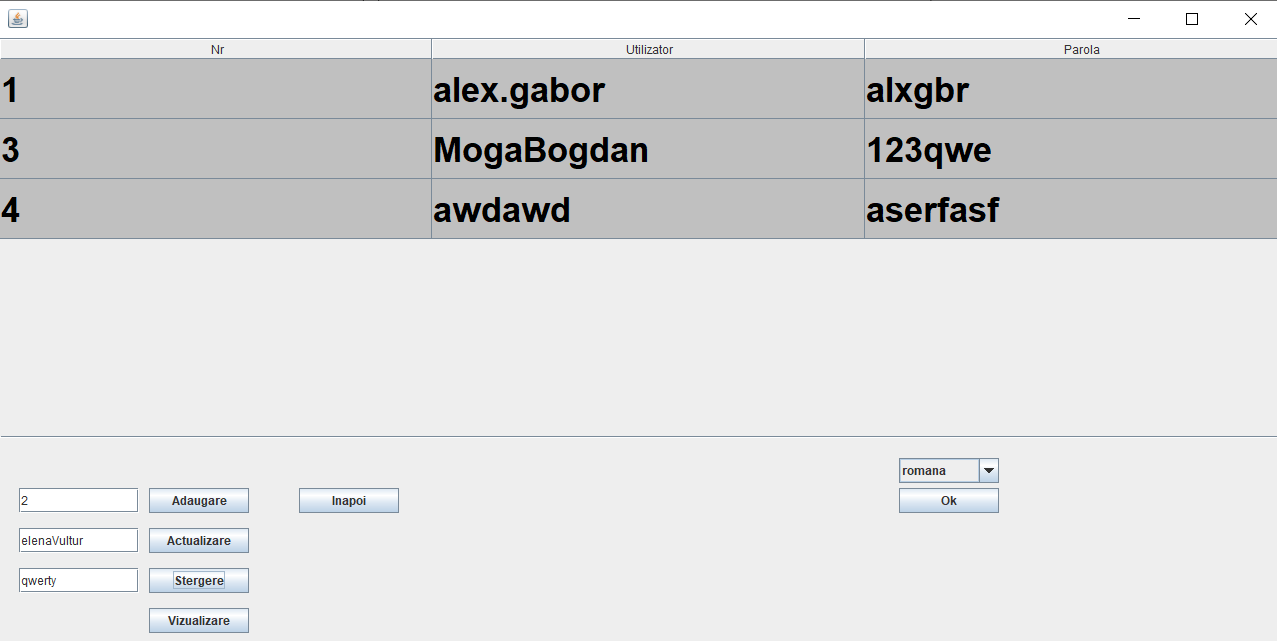


Figură 22-Meniu administrator dupa operatia de adaugare angajat

Dacă dorim sa ștergem un angajat vom selecta angajatul dorit si apoi vom selecta butonul de ștergere.

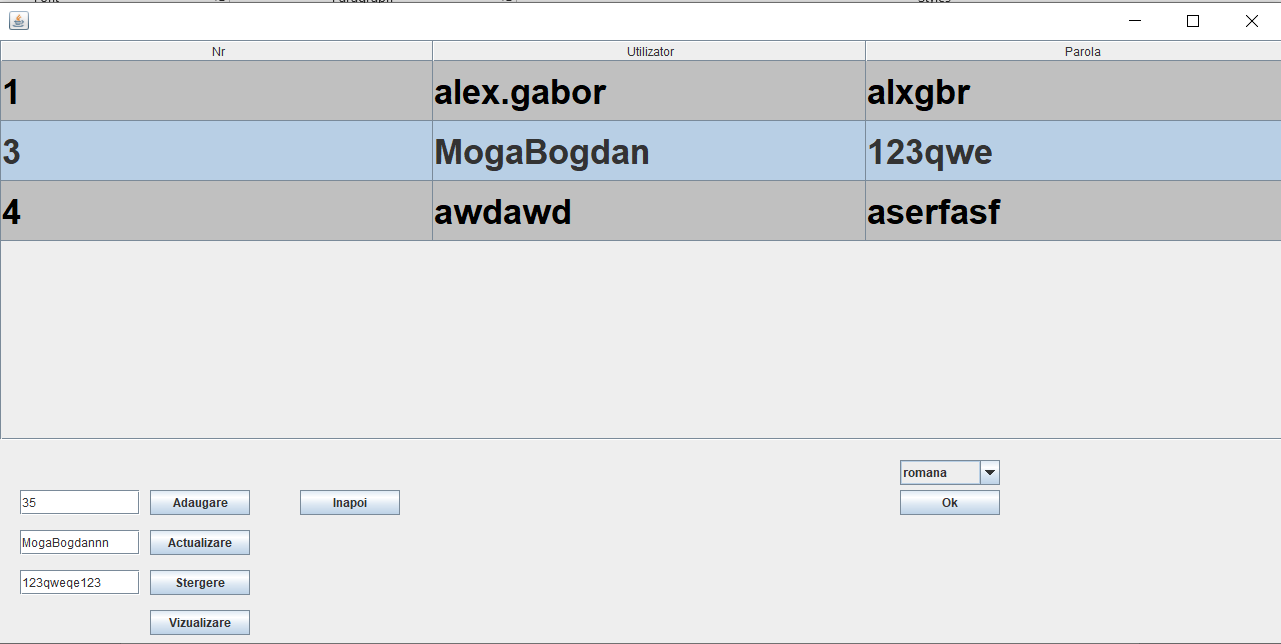


Figură 23-Selectare rand din tabel



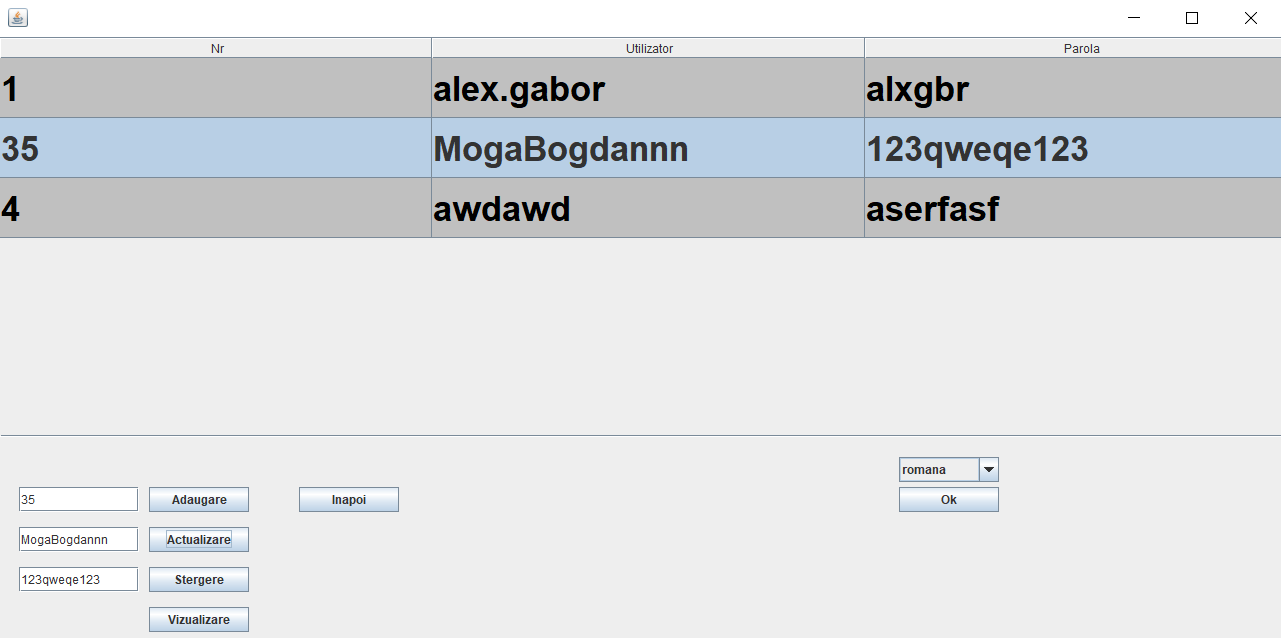
Figură 24-Meniu administrator dupa operatia de stergere

Pentru actualizare se va selecta angajatul care va fi actualizat apoi se vor completa câmpurile de jos cu datele actualizate si se va apăsa butonul de actualizare.



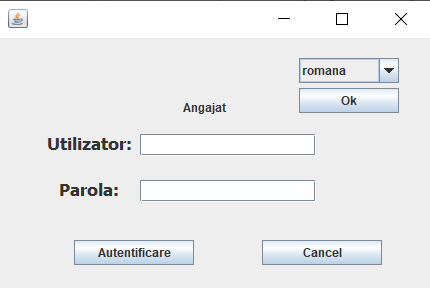
Figură 25-Selectare rand din tabel

Se pot observa in JtextFielduri ca am scris cum vrem sa fie actualizat angajatul respectiv,iar mai jos vom vedea aceste modificări in tabel.

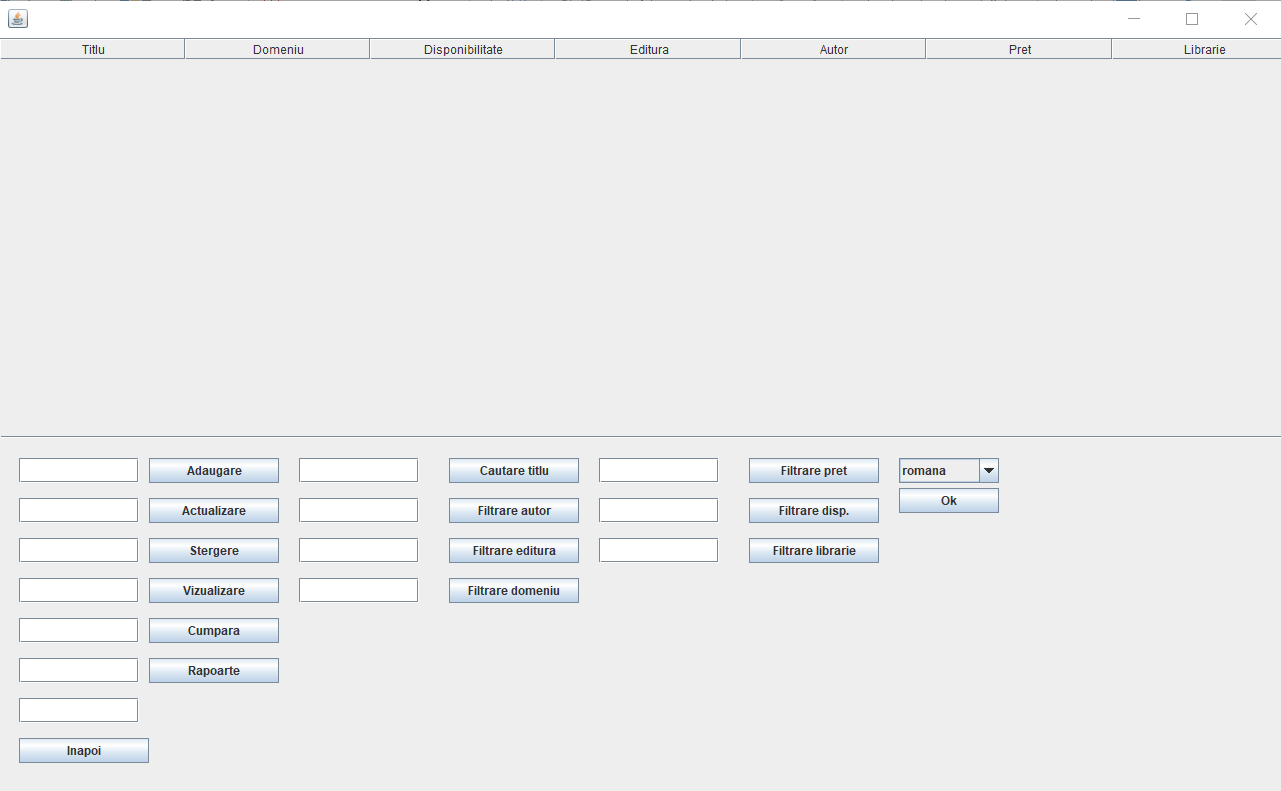


Figură 26-Meniu administrator dupa operatia de actualizare

În cazul în care vrem sa ne autentificăm ca și angajat se va proceda la fel ca si la administrator,vor apărea aceleași mesaje ca si la administrator in funcție de datele introduse.Dacă datele introduse sunt corecte după mesaj se va deschide fereastra angajatului.

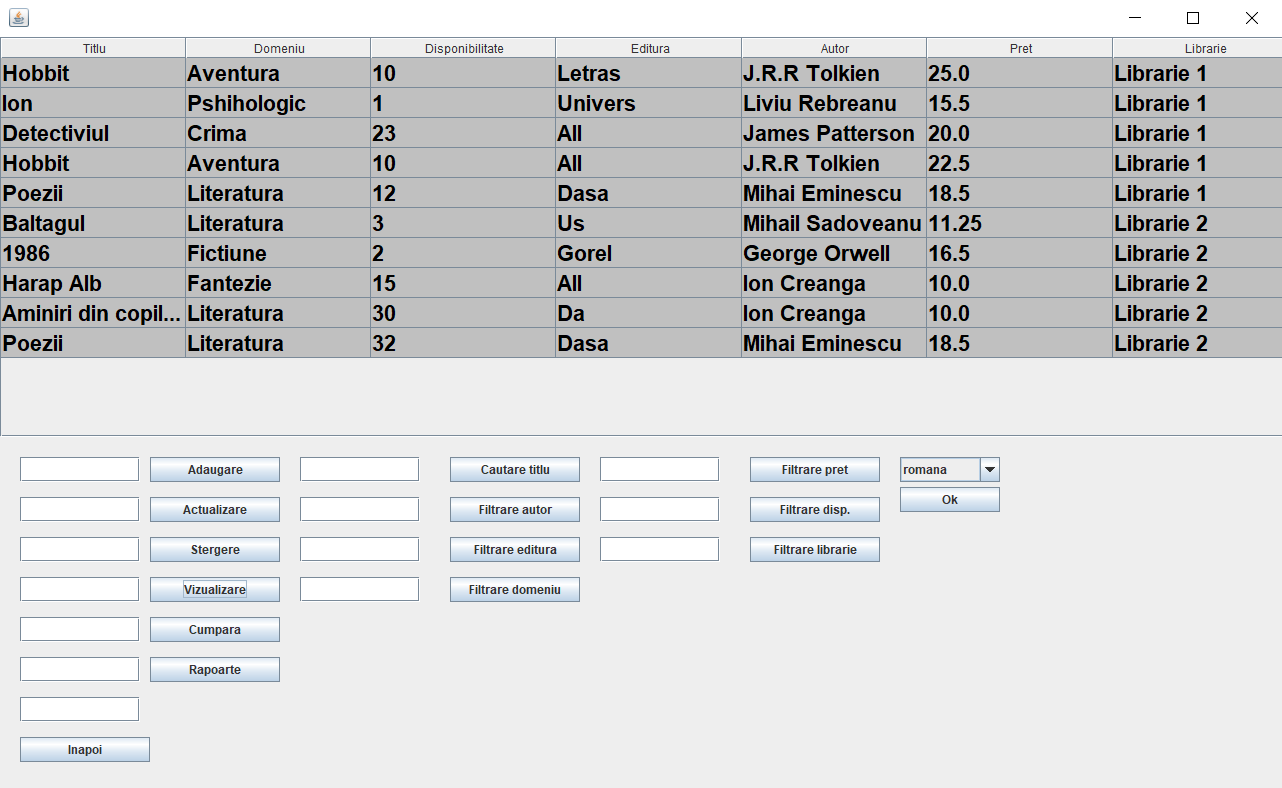


Figură 27-Fereastra login angajat



Figură 28-Meniu angajat

Vizualizarea produselor(cărților) din baza de date.

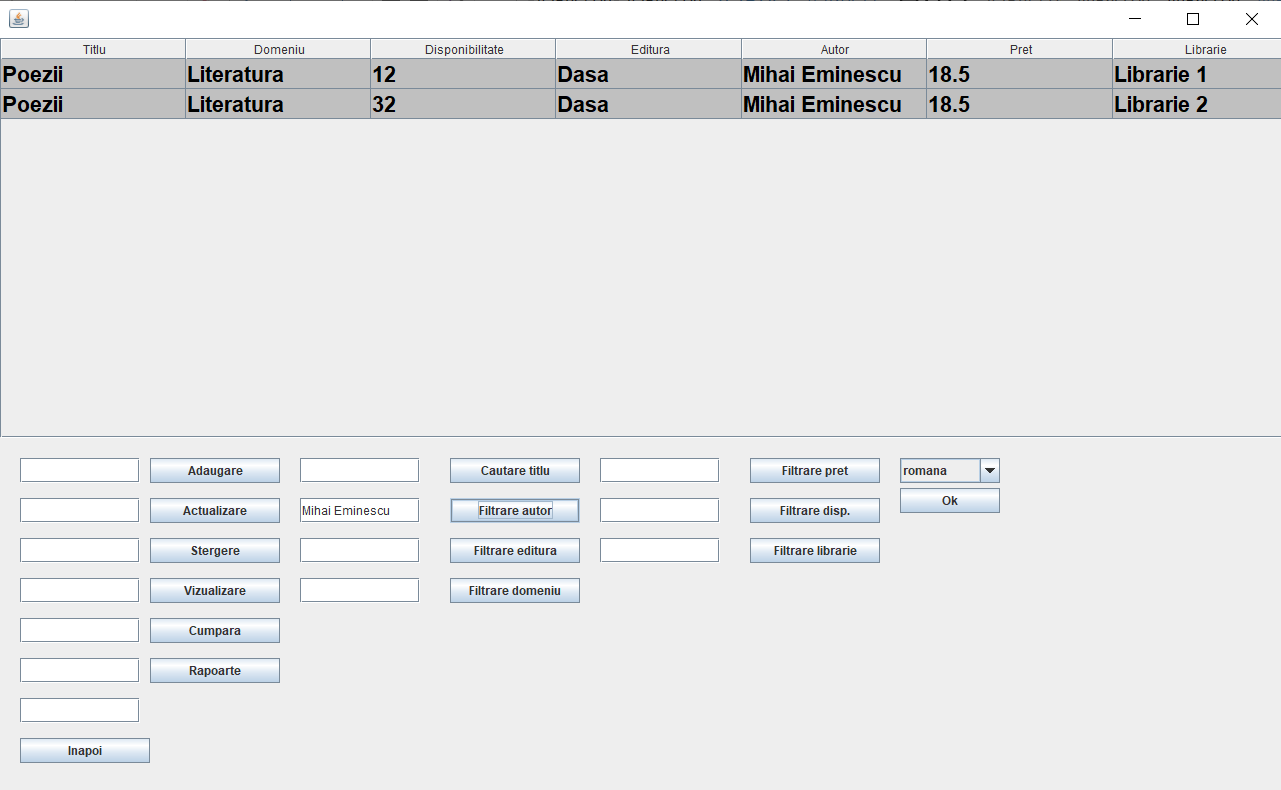


Figură 29-Meniu angajat cu tabelul cartilor

Operațiile CRUD asupra cărților sunt la fel ca si operațiile CRUD asupra angajaților.

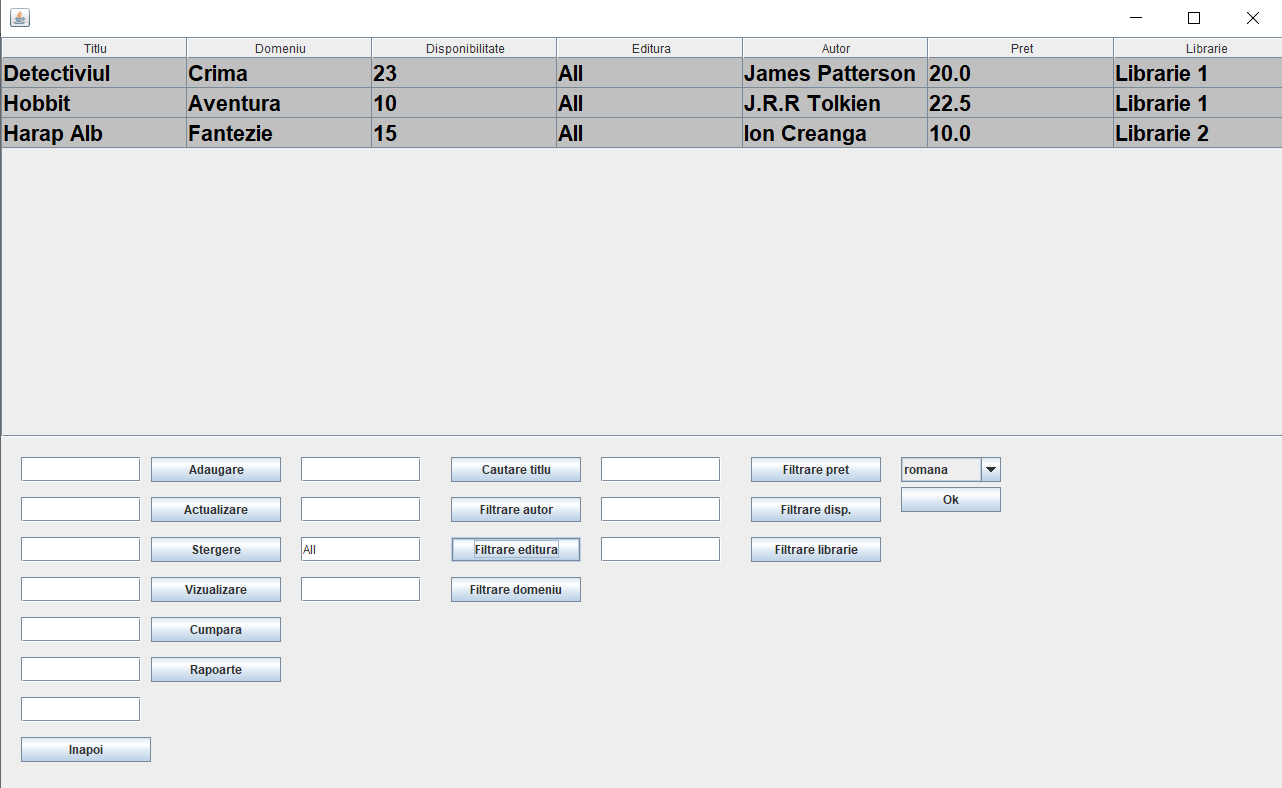
Pentru filtrarea produselor se va completa câmpul de langa butonul corespunzător care indică ce fel de filtrare se poate face.Se poate realiza filtrarea doar dupa un singur criteriu.Aplicația va afisa in tabel doar obiectele care corespund filtraii alese

Filtrare dupa autor



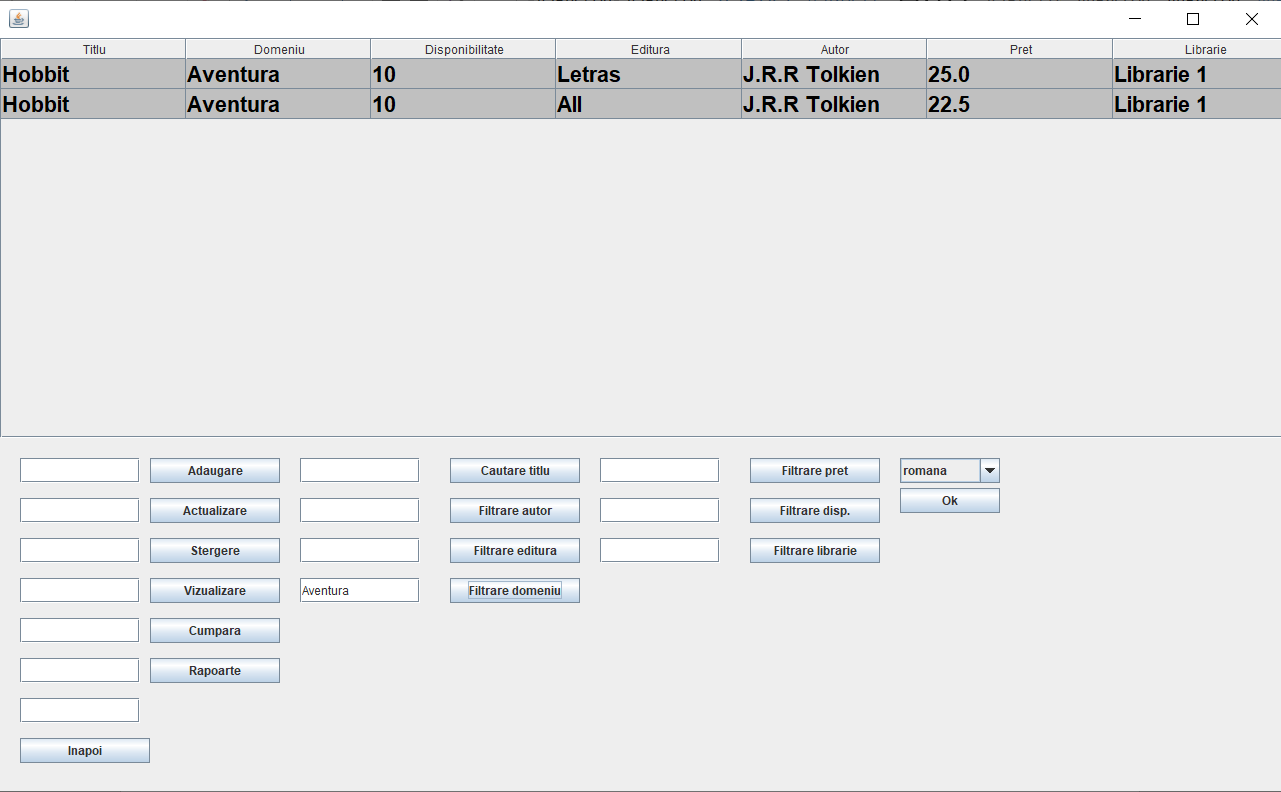
Figură 30-Meniu angajat dupa filtrare autor

Filtrare dupa editura



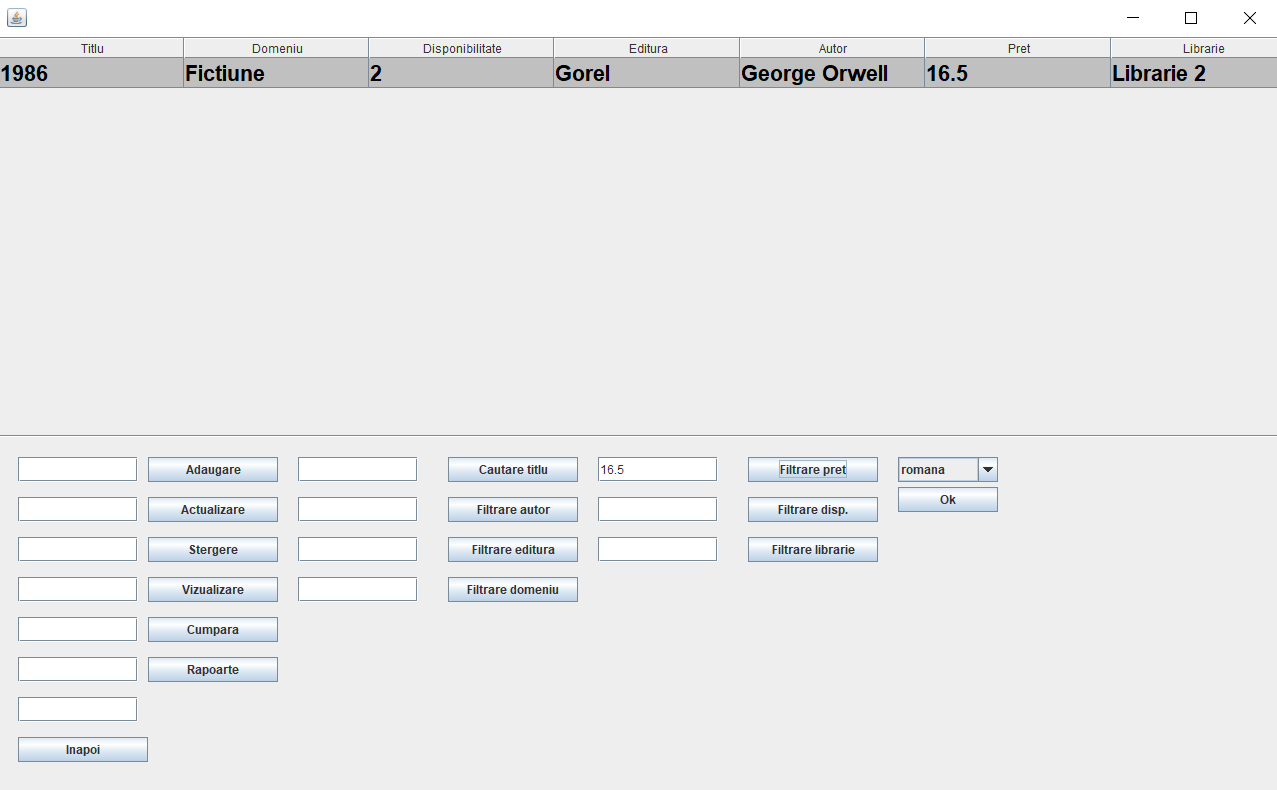
Figură 31--Meniu angajat dupa filtrare editura

Filtrare dupa domeniu



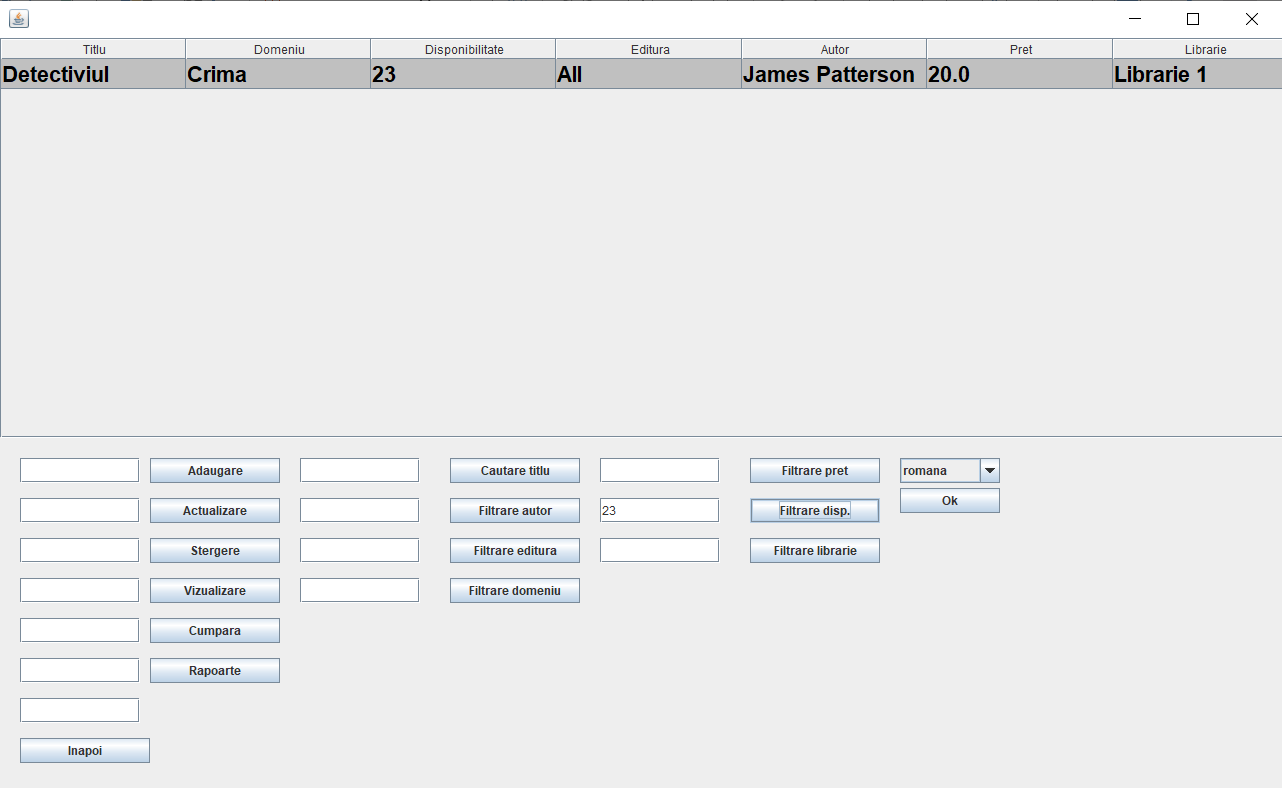
Figură 32--Meniu angajat dupa filtrare domeniu

Filtrare dupa preț



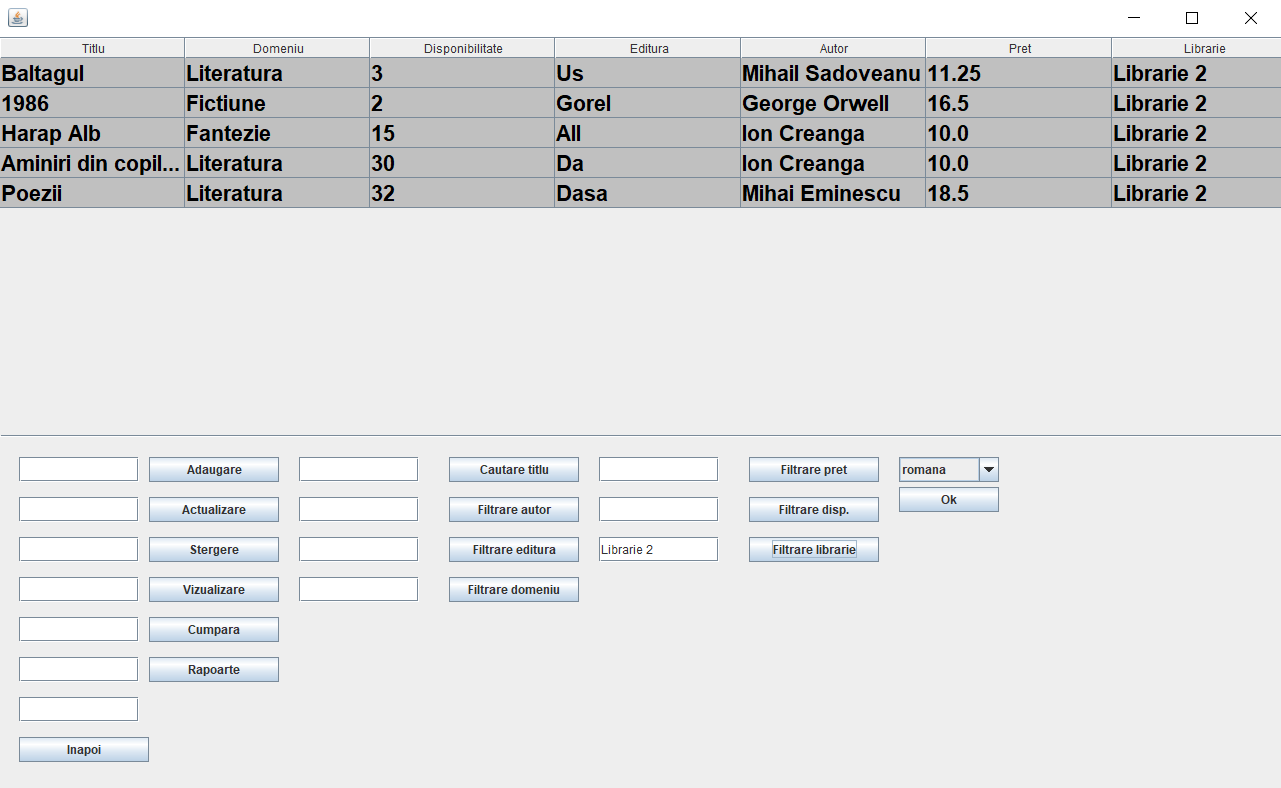
Figură 33--Meniu angajat dupa filtrare pret

Filtrare după disponibilitate



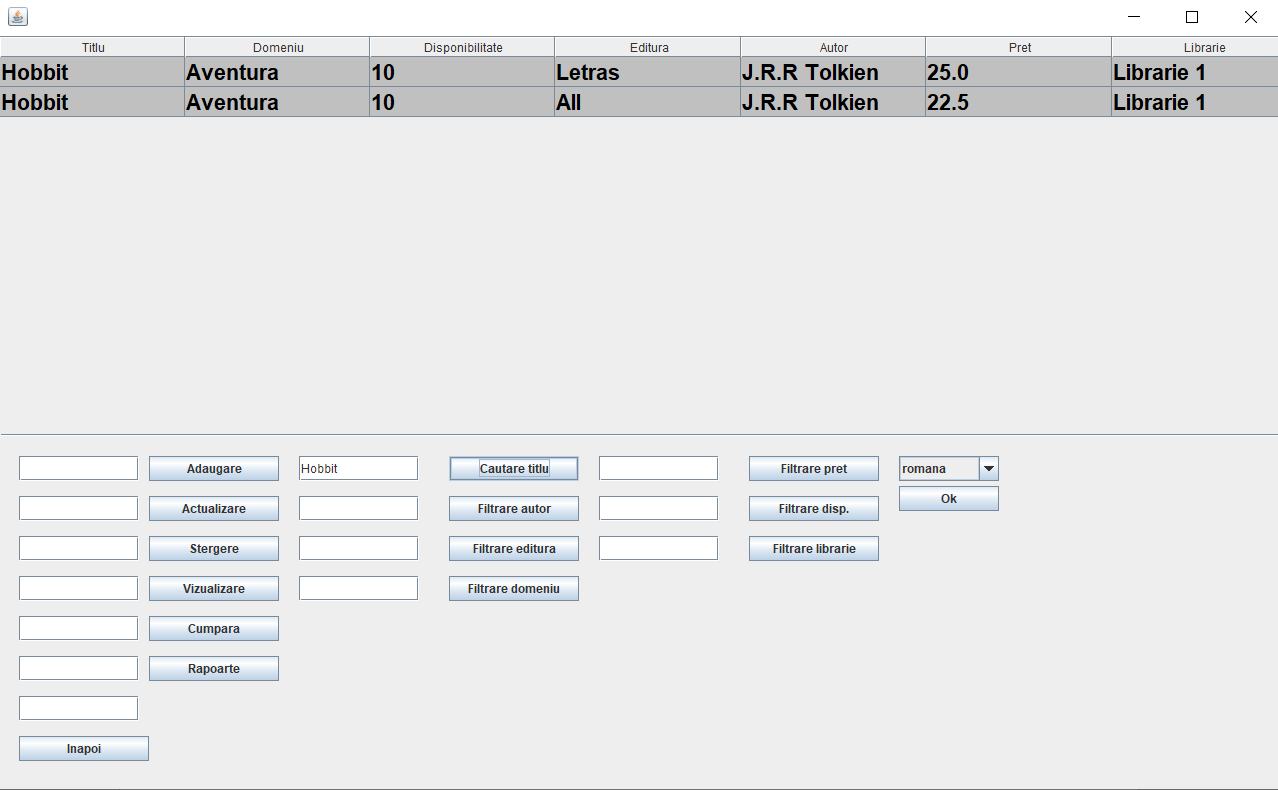
Figură 34--Meniu angajat dupa filtrare disponibilitate

Filtrare după librărie



Figură 35--Meniu angajat dupa filtrare librarie

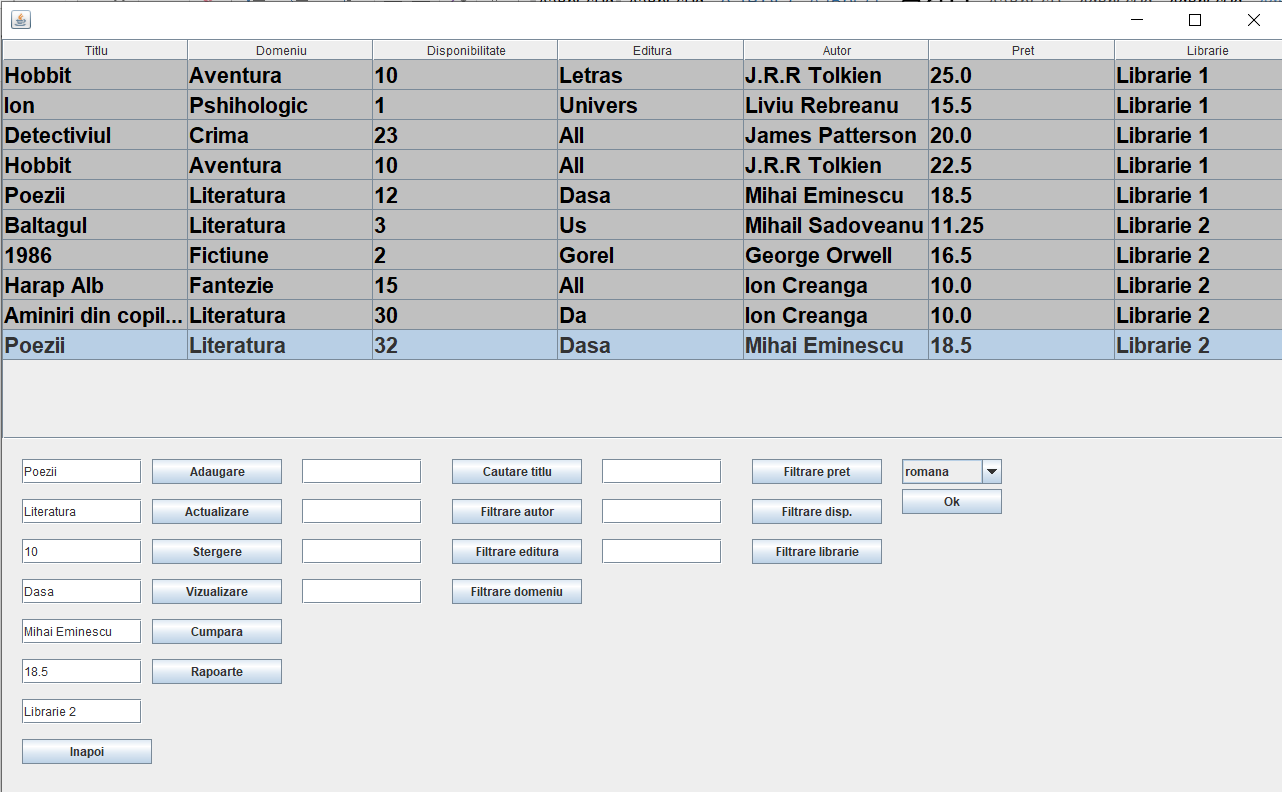
Căutare titlu:Se introduce titlul pe care vrem să îl căutăm si se apasă pe butonul de căutare.Aplicația va returna in tabel elementele care au titlul identic cu cel introdus de angajat.



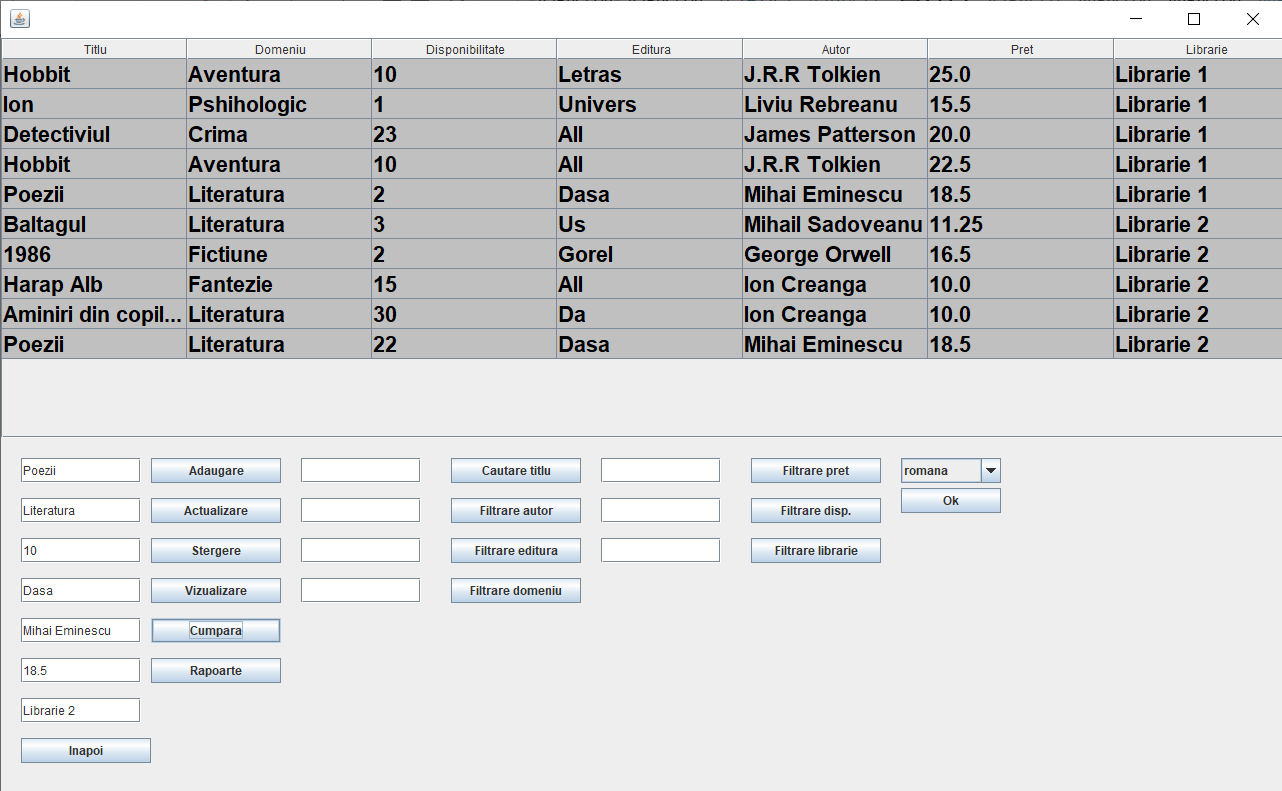
Figură 36-Meniu angajat dupa cautare titlu

Pentru operația de generare de rapoarte.Se apasă butonul de rapoarte,iar elementele din baza de date care pot fi vazute și in tabel in alicație vor fi salvate in cele trei fișiere.Fișierele se gasesc in folderul unde a fost salvat proiectul.

Ultima operație pe care o poate face angajatul este aceea de cumpărare.Se selecteaza un element din tabel,apoi se completeaza in a treia căsuță de sus in jos corespunzătoare disponibilității numărul de exemplare care vor fi cumpărate,iar atunci când se apasăa butonul de cumpărare se va scădea din numărul inițial numărul de elemente cumpărate,iar tabelul va fi actualizat.



Figură 37-Selecrtare rand din tabel



Figură 38-Meniu angajat dupa operatia de cumparare

Operația de cumpărare a fost realizată si cu ajutorul șablonului de design Observer.Acesta notifică atunci când unul sau mai multe exemplare dintr-un produs sunt cumpărate,astfel tabelul se actualizeaza automat si nu mai este nevoie de a apăsa din nou pe butonul de vizualizare.

Ca si celelalte ferestre si aceasta are opțiunea de schimbat limba.