Aplicații integrate pentru întreprinderi

Tema 2

Data publicării: 09.11.2010
Termenul de predare: 30.11.2010

Obiective

Obiectivul general al laboratorului constă în dezvoltarea unui sistem ERP menit să automatizeze munca într-o instituție. În cadrul acestei etape, aplicația va fi modularizată în componente specializate pe anumite operații, fiind capabile să comunice în cadrul unei rețele de calculatoare prin care sunt conectate.

După rezolvarea temei de casă, studentul va fi capabil să:

- proiecteze o aplicație având o structură distribuită pe componente capabile să ruleze pe maşini diferite;
- modeleze comunicația între componentele distribuite ale aplicației;

Cunoştințele necesare pentru rezolvarea temei de casă sunt:

- programarea în limbajul Java;
- manipularea bazelor de date folosind MySQL;
- folosirea mecanismului RMI;

Enunț

Funcționalitatea descrisă la tema precedentă va fi distribuită într-o structură ce va conține trei componente (trei aplicații separate):

- student
- bibliotecar
- arhivă

Accesul la tabelele din baza de date va fi restricționat pentru fiecare componentă în parte astfel:

- student nu va putea accesa (în mod direct) nici o tabelă;
- bibliotecar va putea accesa doar tabela **împrumuturi** având drepturi de citire şi scriere;
- arhivă va putea accesa tabela **volume** cu drepturi de citire şi de scriere;

Din acest motiv, fiecare dintre componente va trebui să pună la dispoziție metode prin care celelalte aplicații să poată accesa informațiile de care ar putea avea nevoie.

1. Un **student**

a. va interoga tabela **volume** prin intermediul componentei **arhiva** apelând metoda ce va avea semnătura:

```
public ArrayList<Volum> cautare_volum (String titlu, String autor);
```

Interogarea propriu-zisă va fi realizată în cadrul componentei **arhiva**, iar rezultatul de tip *ResultSet* va fi trecut într-un obiect de tip ArrayList<Volum> conținând lista cu toate volumele din tabelă care îndeplinesc condițiile precizate.

Observații.

Nu este necesar ca ambii parametri să fie completați, în cazul când doar unul dintre aceştia este `null` sau şirul vid, căutarea făcându-se exclusiv după parametrul specificat.

Puteți adăuga şi alți parametri pentru această metodă, conform implementării de la tema precedentă (spre exemplu: domeniu, editură, loc_apariție, etc).

În cazul în care mai mulți parametri vor fi specificați, combinarea acestora în condiția pentru clauza WHERE din interogarea MySQL corespunzătoare va fi de tipul AND.

Clasa **Volum** va trebui implementată de voi cu toate câmpurile precizate în tabela corespunzătoare în baza de date, va trebui să conțină metode set/get şi trebuie să fie **serializabilă**.

Toate metodele de comunicare între componente vor arunca excepția `java.rmi.RemoteException`.

b. va face cereri către **bibliotecar** pentru împrumut sau pentru restituire volum apelând metode având semnătura:

```
public boolean cerere_imprumut(int cnp, String cota);
public boolean cerere_restituire (int cnp, String cota);
```

Metodele vor întoarce o valoare de tip boolean prin care se va specifica dacă cererea lor a fost sau nu acceptată.

Pentru metoda cerere_imprumut, bibliotecarul va verifica dacă numărul de exemplare disponibile din volumul având cota specificată este disponibil, precum şi faptul că numărul de volume împrumutate de student nu depăşeşte 3, moment în care acceptă cererea, în caz contrar refuzând-o (la fel procedează pentru cazurile student/volum inexistent). În cazul acceptării cererii, bibliotecarul modifică şi numărul de exemplare disponibile pentru volumul respectiv.

Pentru metoda cerere_restituire, bibliotecarul va verifica faptul că volumul a fost împrumutat de student, moment în care acceptă cererea, modificând şi numărul de exemplare disponibile pentru volumul respectiv.

Observație.

Puteți folosi în semnătura metodelor câmpurile pe care le-ați definit drept chei primare în cadrul tabelelor **student**, respectiv **volume**.

2. Un **bibliotecar** va interoga tabela **volume** prin intermediul componentei **arhiva**, apelând metode ce vor avea următoarea semnătură:

```
public boolean adaugare_volum (String cota,
                               String titlu,
                               String autor,
                               String editura,
                               String loc_aparitie,
                               int an,
                               int exemplare_disponibile);
public ArrayList<Volum> cautare_volum (String titlu, String autor);
```

```
public boolean editare_volum  (String cota,
                               String titlu,
                               String autor,
                               String editura,
                               String loc_aparitie,
                               int an,
                               int exemplare_disponibile);
public boolean stergere_volum (String cota);
public int numar_exemplare (String cota, int exemplare_disponibile);
```

Aşadar, bibliotecarul îşi păstrează posibilitatea de a adăuga, modifica şi şterge volume din bibliotecă, însă acum acest lucru se face prin intermediul componentei arhiva care execută propriu-zis interogările.

Metodele respective au un rezultat de tip **boolean** indicând realizarea operației cu succes sau dimpotrivă, datorită unor situații posibile precum:

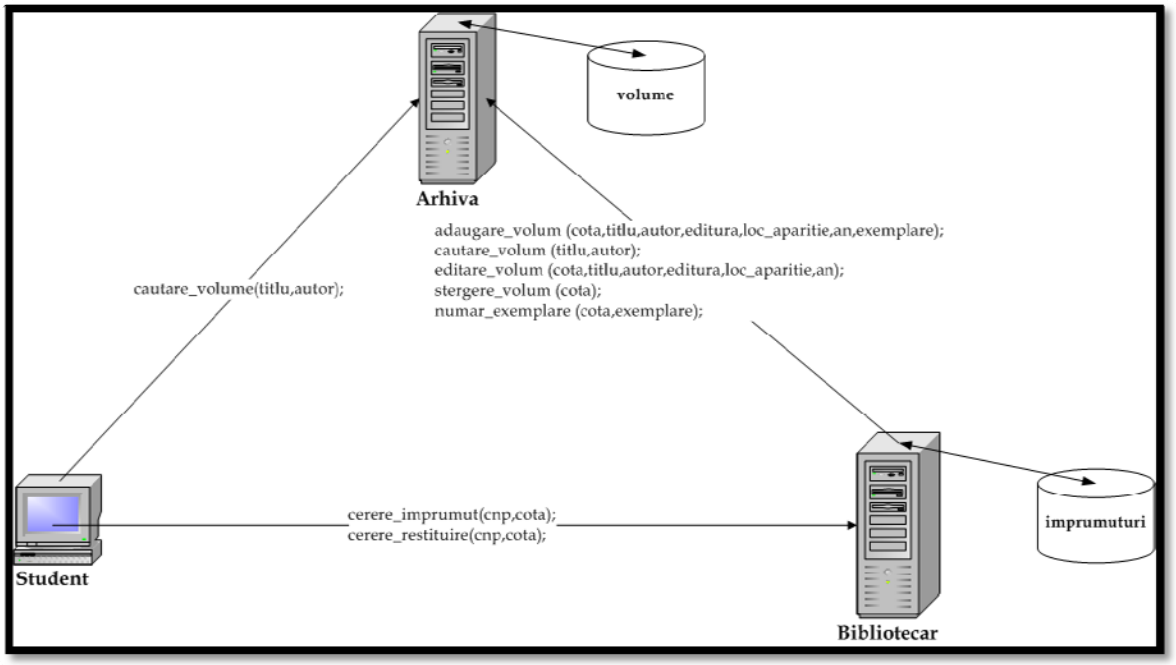
- cota este existentă, titlu şi autor sunt null sau şiruri de caractere vide, an sau exemplare sunt numere negative pentru operația de **adăugare**;
- cota nu există, an este negativ pentru operația de **editare**;

  În cazul în care se doreşte doar editarea unor anumite câmpuri din înregistrarea corespunzătoare volumului având cota specificată, specificarea atributelor pentru care nu se realizează modificarea se va face cu null sau câmpul vid pentru argumentele de tip String, cu 0 pentru argumentele de tip int;
- cota nu există pentru operația de **ştergere**;

Operația număr_exemplare este o metodă de tip set/get care funcționează astfel:

- dacă se doreşte **stabilirea** numărului de exemplare pentru volumul specificat prin cotă, parametrul exemplare_disponibile este >=0 iar rezultatul este 0 sau 1 după cum operația a fost sau nu realizată cu succes;
- dacă se doreşte **obținerea** numărului de exemplare pentru volumul specificat prin cotă, parametrul exemplare disponibile este -1, iar rezultatul este numărul de exemplare corespunzător.

Un rezultat negativ va indica producerea unei erori (cotă inexistentă).



Arhitectura distribuită pe componente a aplicației

Observații generale.

După cum veți observa pe parcursul implementării temei, arhitectura descrisă nu cuprinde toate metodele necesare interacțiunii dintre componente. Voi veți alege ce alte metode vor mai pune la dispoziție componentele.

Metodele descrise mai sus nu trebuie să aibă în mod obligatoriu semnătura care a fost precizată, ea poate fi adaptată conform implementării voastre de la tema precedentă însă având aceeaşi funcționalitate.

Va trebui să specificați pentru fiecare componentă şi politicile de securitate corespunzătoare cu care vor fi apelate.

Aşa cum a fost reprezentat în figura de mai sus, studentul are comportament de client, arhiva are comportament de server iar bibliotecarul comportament mixt, atât de client cât şi de server.

Pentru această temă vom lucra cu două tabele (volume şi împrumuturi) din baza de date care, aşa cum reiese şi din figură, a fost şi ea distribuită[1].

Limbajul de programare în care trebuie se va implementa tema este Java, baza de date este MySQL, iar comunicarea între componente se bazează pe tehnologia RMI.

Modul de punctare al temei este următorul:
1p – definirea interfețelor de comunicare;
1p – componenta student;
3p – componenta bibliotecar;
4p – componenta arhivă;
1p – explicarea corespunzătoare a soluției prin README şi comentarii în cod, utilizarea de nume sugestive pentru variabile şi metode
1p – bonus pentru prezentarea temei înainte de termenul limită;

Condiții de realizare şi predare

Tema va fi realizată individual în cadrul laboratorului şi va fi prezentată pe 30.11.2010 în intervalul orar pe care l-ați ales.

La fiecare laborator veți realiza o anumită parte a temei:

- Laborator 05 (09.11.2010) – dezvoltarea interfețelor de comunicare între student şi arhivă, între student şi bibliotecar, între bibliotecar şi arhivă;
- Laborator 06 (16.11.2010) – implementarea componentei arhiva;
- Laborator 07 (23.11.2010) – realizarea componentelor student/bibliotecar, precum şi a unor scripturi care să ilustreze diferite comportamente.

După prezentarea temei, va trebui să incarcati pe site o arhivă de tip .zip (cu numele Grupa34XCX_NumePrenume_Tema2.zip) care să conțină scriptul[2] pentru crearea tabelelor din baza de date (numele bazei de date trebuie să aibă forma Grupa34XCX_ NumePrenume), sursele aplicației, scripturi[3] spre a ilustra diferite comportamente (adăugare, editare, stergere volume, cerere imprumut şi restituire) precum şi un README în care să explicați soluția aleasă.

---

[1] Există sisteme de gestiune pentru baze de date distribuite care pot asigura o arhitectură de acest tip, corespunzând unei tehnici de distribuire prin fragmentare verticală.

[2] Scriptul pentru crearea tabelelor din baza de dat va fi trimis în următoarele situații:

- structura tabelelor a fost modificată față de cea de la tema precedentă;
- la tema precedentă nu au fost inclusă în script şi popularea tabelelor cu date.

[3] Scripturile vor conține lansări în execuție ale aplicației cu diferiți parametri.

Încarcarea pe site nu este redundantă, temele vor fi comparate prin aplicații specializate pentru a se depisata eventualele fraude. În această situație, atât sursa cât şi destinația vor fi punctate cu 0.