Aplicații integrate pentru întreprinderi

Tema 3

Data publicării: 30.11.2010
Termenul de predare: 21.12.2010

Obiective

Obiectivul general al laboratorului constă în dezvoltarea unui sistem ERP menit să automatizeze munca într-o instituție. În cadrul acestei etape, componentele student şi bibliotecar (implementate în cadrul temei precedente) vor avea un comportament de tipul *publisher-subscriber*, comunicația între ele realizându-se prin intermediul unei alte componente care va avea rolul unei cozi de mesaje.

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 o comunicație (asincronă) între componentele distribuite ale aplicației folosind mecanisme de nivel scăzut;

Cunoştințele necesare pentru rezolvarea temei de casă sunt:

- programarea în limbajul Java;
- manipularea bazelor de date folosind MySQL;
- folosirea de socketi TCP pentru comunicația între procese;

Enunț

Se va păstra funcționalitatea de la tema precedentă (componentele student, bibliotecar şi arhivă vor pune la dispoziție aceleaşi metode), însă studentul şi bibliotecarul nu vor mai rula (neapărat) simultan[1], comunicația între cele două componente fiind asincronă. Interacțiunea dintre student şi bibliotecar va fi realizată prin intermediul unei componente suplimentare ***registru*** care înregistrează cererile de împrumut sau de restituire din partea studenților precum şi deciziile pe care bibliotecarii le iau în privința acestor cereri.

Observație

Doar componenta bibliotecar îşi pierde comportamentul de tip server, componenta arhiva va rula permanent, primind cereri de la student şi bibliotecar pe care le va soluționa imediat.

În continuare, accesul la tabelele din baza de date este restricționat precum s-a precizat în tema precedentă.

Comunicația între cele patru componente se va realiza folosind mecanisme de nivel scăzut, şi anume **socketi TCP**.

---

[1] Datorită crizei financiare, numărul de bibliotecari a fost redus substanțial şi programul de lucru al bibliotecarilor rămaşi nu mai acoperă întreaga perioadă cât biblioteca este deschisă. Studenții trebuie să lase cereri către bibliotecari, acestea fiind soluționate de un bibliotecar atunci când ajunge la serviciu.

1. Un **student**
a. poate interoga tabela **volume** prin intermediul componentei **arhiva** transmițând un mesaj (şir de caractere) având următoarea structură:

cautare_volum
titlu
autor

Separarea informațiilor în structura mesajului se face prin intermediul caracterului **"*"** (se presupune că acest caracter nu se regăseşte în nici una dintre componentele elementelor care alcătuiesc şirul de caractere transmis[2]).

Exemplu:
```
cautare_volum*Poezii*Mihai Eminescu
```

Răspunsul va fi transmis imediat de componenta **arhiva** sub forma unui mesaj (şir de caractere) conținând toate înregistrările care respectă criteriile specificate în interogare:

raspuns_cautare_volum							
număr_înregistrări (n)	$cota_1$	$titlu_1$	$autor_1$	$editura_1$	$loc\_aparitie_1$	an_1	$exemplare\_disponibile_1$
	$cota_2$	$titlu_2$	$autor_2$	$editura_2$	$loc\_aparitie_2$	an_2	$exemplare\_disponibile_2$
	...	...	...	...	...	...	...
	$cota_n$	$titlu_n$	$autor_n$	$editura_n$	$loc\_aparitie_n$	an_n	$exemplare\_disponibile_n$

Separarea între înregistrările din baza de date se face – la fel ca mai sus – prin intermediul caracterului **"*"**, iar între atributele ce constituie înregistrarea folosind caracterul **"#"**.

Exemplu[3]:
```
raspuns_cautare_volum*2*
LIT000001#Poezii#Mihai Eminescu#Universul Cartii#Bucuresti#1978#5*
LIT000002#Poezii si corespondenta#Mihai Eminescu#Minerva#Iasi#1995#0
```

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 (domeniu, editură, loc_apariție, an, exemplare_disponibile).

În cazul în care mai mulți parametri vor fi specificați, combinarea acestora în clauza WHERE din interogarea MySQL corespunzătoare poate fi AND sau OR.

---

[2] În situația în care totuşi acest caracter se regăseşte în informațiile reținute în baza de date, puteți folosi alt caracter pentru a delimita elementele din structura mesajului.

[3] Mesajul nu va conține caractere de tip linie nouă (`'\n'`), în exemplu şirul de caractere răspuns este pe mai multe linii pentru a scoate în evidență structura sa.

b. va face cereri către **bibliotecar** prin intermediul componentei **registru** pentru împrumut sau restituire volum prin mesaje având următoarea structură:

imprumut_volum
CNP
cota

restituire_volum
CNP
cota

Structura mesajelor va avea elementele separate

Exemple:
```
imprumut_volum*1880219320098*LIT000001
restituire_volum*1891008610002*LIT000002
```

Componenta **registru** va reține astfel de cereri într-o structură tip coadă de mesaje (*eng.* message queue) şi le va transmite către componenta **bibliotecar** spre a fi soluționate, atunci când îi vor fi solicitate.

Pentru reținerea informațiilor cu privire la cereri (şi modul în care au fost soluționate) se poate folosi o structură de tip `HashMap<Cerere,Rezultat>` unde clasa `Cerere` reține informații cu privire la tipul cererii (imprumut, restituire), CNP şi cota. Obiectul de tip `Rezultat` va oferi date referitoare la soluționarea cererii (0 = neanalizată, -1 = respinsă, 1 = acceptată).

De asemenea, **studentul** va avea posibilitatea de a interoga componenta **registru** cu privire la rezultatul cererilor pe care le-a realizat anterior.

rezultat_cereri
CNP

Exemplu:
```
rezultat_cereri*1900721610054
```

Răspunsul va consta într-un mesaj care conține soluționările date de bibliotecari pentru toate cererile formulate. În momentul în care componenta **arhiva** transmite o astfel de informație către componenta **arhiva** va şterge din obiectul (intern) de tip coadă de mesaje toate cererile ***care au fost soluționate*** şi care au fost incluse în şirul de caractere transmis[4].

raspuns_rezultat_cereri			
număr_înregistrări (n)	tip_cerere_1	$cota_1$	$rezultat_1$
	tip_cerere_2	$cota_2$	$rezultat_2$
	...	...	...
	tip_cerere_n	$cota_n$	$rezultat_n$

[4] Mesajul către **student** va include ***toate cererile*** care au fost formulate (deci inclusiv cele care nu au fost încă analizate de **bibliotecar**), însă doar cele care au fost deja soluționate vor fi eliminate de componenta **arhivă** din structura internă după ce au fost transmise studentului.

Delimitatorii între informațiilor din cadrul mesajului vor fi, ca mai sus, caracterele **"*"** şi **"#"**.

Exemple[5]:

```
raspuns_rezultat_cereri*4*
imprumut#LIT000001#0*
restituire#LIT000002#-1*
imprumut#LIT000003#1*
restituire#LIT000004#0
```

Observație

Puteți folosi drept informații în mesajele schimbate între student şi arhivă respectiv student şi registru câmpurile folosite drept chei primare în cadrul tabelelor **student**, respectiv **volume**.

2. Un **bibliotecar**

a. va interoga tabela **volume** prin intermediul componentei **arhiva**, transmițând mesaje care vor avea următoarea structură:

a1. adăugare volum

adaugare_volum
cota
titlu
autor
editura
loc_apariție
an
exemplare_disponibile

Exemplu:

```
adaugare_volum*LIT123456*Amintiri din copilarie*Ion Creanga*
Polirom*Bucureşti*2005*10
```

Răspunsul la o astfel de cerere are structura:

raspuns_adaugare_volum
rezultat

unde rezultat poate avea valorile `succes` sau `insucces`[6].

Exemplu:

```
raspuns_adaugare_volum*succes
```

[5] Cererile care vor fi eliminate din obiectul de tip coadă de mesaje a componentei **arhiva** vor fi, pentru exemplul dat, `restituire#LIT000002#-1` şi `imprumut#LIT000003#1` pentru că acestea au fost deja soluționate de bibliotecari.

[6] Cauze care pot determina un răspuns de tip `insucces` sunt: cotă existenta, titlu şi autor necompletate, an sau exemplare_disponibile de tip şir de caractere sau numere negative.

a2. căutare volum

Mesajele transmise între componenta **biblioteca** şi componenta **arhiva** au acceaşi structură cu mesajele dintre componanta **student** şi componenta **arhiva** care vizează aceeaşi funcționalitate.

a3. editare volum

editare_volum
cota
titlu
autor
editura
loc_apariție
an
exemplare_disponibile

Exemplu:
```
editare_volum*LIT987654*Fratii Jderi*Mihail Sadoveanu*
Cartea Romaneasca*Cluj-Napoca*1978*5
```

Răspunsul la o astfel de cerere are structura:

raspuns_editare_volum
rezultat

unde rezultat poate avea valorile `succes` sau `insucces`[7].

Exemplu:
```
raspuns_editare_volum*insucces
```

a4. ştergere volum

stergere_volum
cota

Exemplu:
```
stergere_volum*LIT111222
```

Răspunsul la o astfel de cerere are structura:

raspuns_stergere_volum
rezultat

unde rezultat poate avea valorile `succes` sau `insucces`[8].

Exemplu:
```
raspuns_stergere_volum*insucces
```

[7] Cauze care pot determina un răspuns de tip `insucces` sunt aceleaşi ca în cazul operației de adaugare volum (cu excepția faptului că pentru operația de editare cota trebuie să fie existentă).
[8] Cauze care pot determina un răspuns de tip `insucces` sunt cota inexistentă.

a5. număr exemplare

numar_exemplare
cota
exemplare_disponibile

unde exemplare_disponibile poate lua următoarele valori:

- -1, caz în care se dorește obținerea numărului de exemplare disponibile pentru cota specificată;
- o valoare pozitivă (>=0), caz în care se dorește stabilirea numărului de exemplare disponibile pentru cota specificată;

Exemple:

```
numar_exemplare*LIT111111*-1
numar_exemplare*LIT222222*10
```

Răspunsul la o astfel de cerere are structura:

raspuns_numar_exemplare
rezultat

unde rezultat poate avea următoarele valori:

- o valoare pozitivă (>=0) indicând numărul de exemplare disponibile pentru cota specificată;
- -1 – operația de stabilire a numărului de exemplare a fost făcută la nivelul bazei de date
- -2 – operația de stabilire a numărului de exemplare nu a reuşit

Exemple:

```
raspuns_numar_exemplare*25
raspuns_numar_exemplare*-2
```

b. va soluționa cererile care au fost realizate de către studenți şi care se găsesc în coada de aşteptare a componentei **registru** pe care o interoghează:

cereri_in_asteptare

Exemplu:

```
cereri_in_asteptare
```

Răspunsul la o astfel de interogare este o listă cuprinzând cererile realizate de studenți şi care nu au fost soluționate.

raspuns_cereri_in_asteptare			
număr_înregistrări (n)	tip_cerere_1	CNP_1	$cota_1$
	tip_cerere_2	CNP_2	$cota_2$
	...	...	...
	tip_cerere_n	CNP_n	$cota_n$

Exemplu:

```
raspuns_cereri_in_asteptare*2*
imprumut#1870427012003#LIT000001*
returnare#1890526102304#LIT000002
```

Bibliotecarul va încerca să soluționeze fiecare cerere în parte, transmițând răspunsul către componenta registru.

Pentru cererea de împrumut, 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 cererea de restituire, bibliotecarul va trebui să verifice faptul că volumul a fost împrumutat de student, moment în care acceptă cererea, modificând şi numărul de exemplare disponibile pentru volumul respectiv[9].

Soluționarea unei cereri este transmisă componentei arhiva sub forma unui mesaj având structura:

solutionare_cerere
tip_cerere
CNP
cota
rezultat

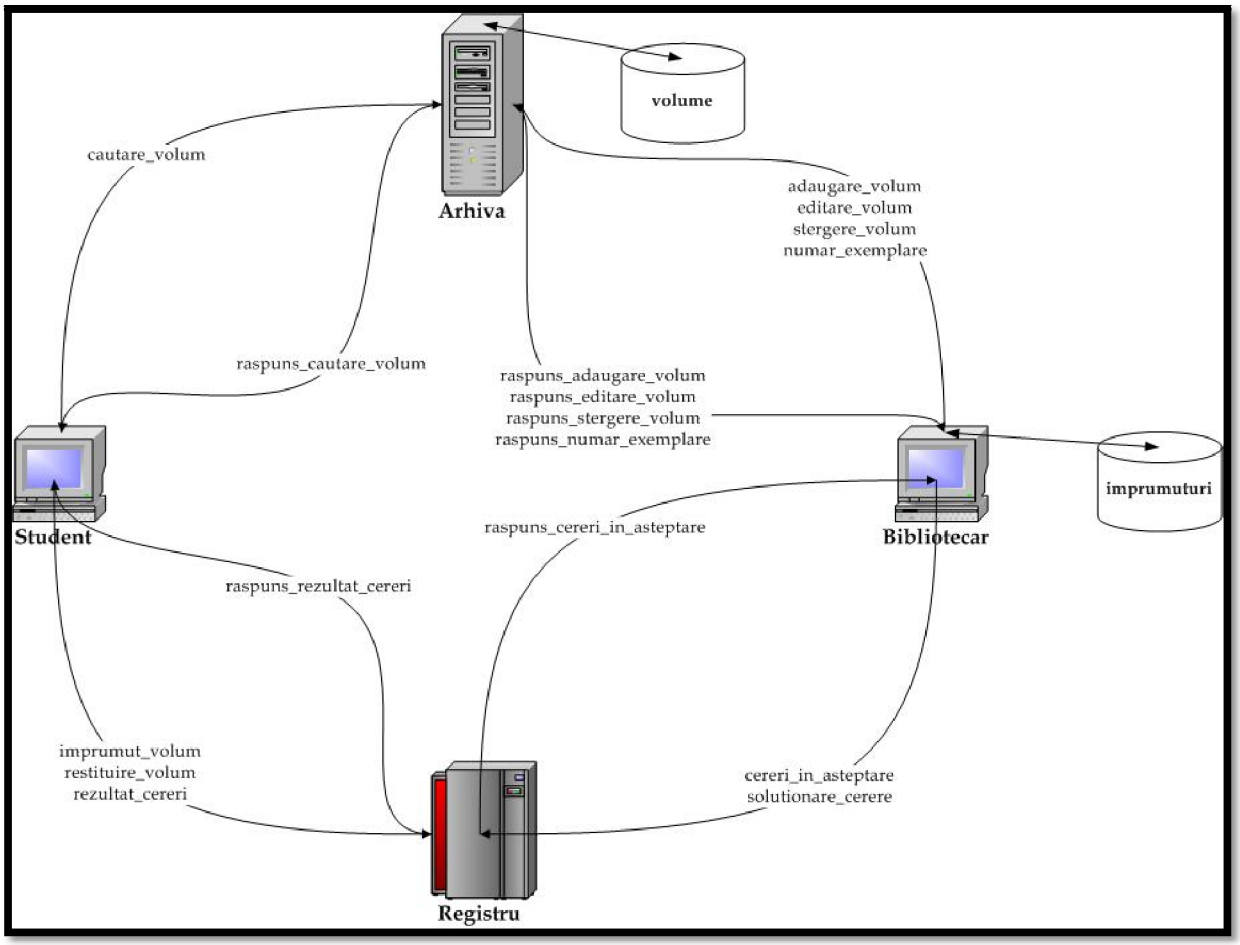
unde rezultat poate avea valorile `succes` sau `insucces`.

Exemple:

```
solutionare_cerere*imprumut*1870427012003*LIT000001*succes
solutionare_cerere*returnare*1890526102304*LIT000002*insucces
```

În momentul în care este primit, rezultatul va fi marcat în structura de tip coadă de aşteptare de către componenta **arhiva**.

[9] Toate aceste operații vor fi realizate prin intermediul unui schimb de mesaje corespunzător între componenta **bibliotecă** şi componenta **arhivă**.



Arhitectura distribuită pe componente a aplicației şi mesajele transmise între componente

#	Mesaj	Sursă	Destinație
1	cautare_volum	student/bibliotecar	arhiva
2	adaugare_volum	bibliotecar	arhiva
3	editare_volum	bibliotecar	arhiva
4	stergere_volum	bibliotecar	arhiva
5	numar_exemplare	bibliotecar	arhiva
6	cereri_in_asteptare	bibliotecar	registru
7	solutionare_cerere	bibliotecar	registru
8	imprumut_volum	student	registru
9	restituire_volum	student	registru
10	rezultat_cereri	student	registru
11	raspuns_rezultat_cereri	registru	student
12	raspuns_cereri_in_asteptare	registru	bibliotecar
13	raspuns_cautare_volum	arhiva	student/bibliotecar
14	raspuns_adaugare_volum	arhiva	bibliotecar
15	raspuns_editare_volum	arhiva	bibliotecar
16	raspuns_stergere_volum	arhiva	bibliotecar
17	raspuns_numar_exemplare	arhiva	bibliotecar

Observații generale

Mesajele descrise mai sus nu trebuie să aibă în mod obligatoriu structura care a fost precizată, aceasta se poate adapta conform particularităților de la temele precedente dar având aceeaşi funcționalitate. Totodată, puteți specifica mesaje suplimentare în funcție de operațiile pe care doriți să le implementați.

Socketii vor fi creați de componentele **arhiva** şi **registru**, iar comunicația dintre student şi bibliotecar şi aceste componente va fi încheiată printr-un mesaj (şir de caractere predefinit) cum ar fi `incheiere_comunicatie`.

Comportamentele componentelor **student**, **registru** şi **bibliotecar** este mixt, fiecare dintre acestea este simultan *publisher* şi *subscriber* în raport cu diferite operații şi componente.

Şi pentru această temă vom lucra doar cu cele două tabele din baza de date (volume şi împrumuturi) ca şi la tema precedentă.

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

Barem de notare

2p – realizarea comunicației între student şi arhivă, între bibliotecar şi arhivă;
2p – proiectarea componentei registru;
2p – implementarea comunicației între student şi registru;
2p – implementarea comunicației între bibliotecar şi registru;
2p – modularizarea aplicației, explicarea soluției în fişierul README şi comentarii în cod, utilizarea de nume sugestive pentru variabile şi metode;
1p – bonus pentru prezentarea temei cu cel puțin o săptămână înainte de termenul limită;
1p – bonus pentru sincronizarea accesului la resurse la nivelul componentei registru;

Condiții de realizare şi predare

Tema va fi realizată individual în cadrul laboratorului şi va fi prezentată pe 21.12.2010 în intervalul orar pe care l-ați ales.

La fiecare laborator veți realiza o anumită parte a temei:

- Laborator 09 (30.11.2010) – realizarea comunicației între student şi arhivă între bibliotecar şi arhivă folosind socketi TCP;
- Laborator 10 (07.12.2010) – proiectarea componentei registru;
- Laborator 11 (14.12.2010) – implementarea comunicației student/registru, bibliotecar/registru (folosind socketi TCP) 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_Tema3.zip) care să conțină scriptul pentru crearea tabelelor din baza de date (numele bazei de date trebuie să aibă forma Grupa34XCX_ NumePrenume), sursele aplicației, scripturi[10] spre a ilustra diferite comportamente (adăugare, editare, stergere volume, cerere imprumut şi restituire) precum şi un README în care să explicați soluția aleasă.

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.

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