Gestionarea unei biblioteci

Adrian Manea, 510

12 ianuarie 2020

Cuprins

DE ADĂUGAT/CLARIFICAT

SECȚIUNEA 1

DESCRIEREA APLICAȚIEI

Aplicația servește la gestionarea unei biblioteci, în relația cu cititorii, oferind roluri speciale pentru clienți din mediul academic.

1.1 Utilizatori și atribute

Utilizatorii aplicației, identificați prin IP-uri specifice sînt:

- **Administratori** ai bazei de date;
- **Personal educațional** (`EDU`);
- **Cititori** din afara mediului educațional (`NEDU`);
- **Bibliotecari**;
- **Cititori înregistrați, dar neabonați** (`RESTUL`).

Considerăm că studenții și profesorii fac parte din personalul educațional (Edu), statut pe care trebuie să-l verifice periodic. De asemenea, un cititor oarecare poate deveni Edu în timp, dacă adaugă verificarea.

1.2 Procesele din aplicație

(P1) Vizualizarea cărților în stoc;

(P2) Vizualizarea bibliotecarilor, cu specializările lor;

(P3) Adăugarea unei cărți;

(P4) Adăugarea unui cititor;

(P5) Adăugarea unui abonat Edu;

(P6) Înregistrarea unui cont public;

(P7) Actualizarea stocului unor cărți;

(P8) Verificarea statutului Edu;

(P9) Administrarea abonamentului;

(P10) Vizualizarea cărților de o anumită specializare;

(P11) Vizualizarea revistelor de o anumită specializare;

(P12) Adăugare reviste;

(P13) Administrare abonament revistă;

(P14) Administrare abonament la newsletter;

(P15) Împrumut carte;

(P16) Feedback bibliotecar.

De exemplu, **descompunerea funcțională** a procesului (P14), de administrare a abonamentului la o revistă, poate fi reprezentat ca în figura 1.1.

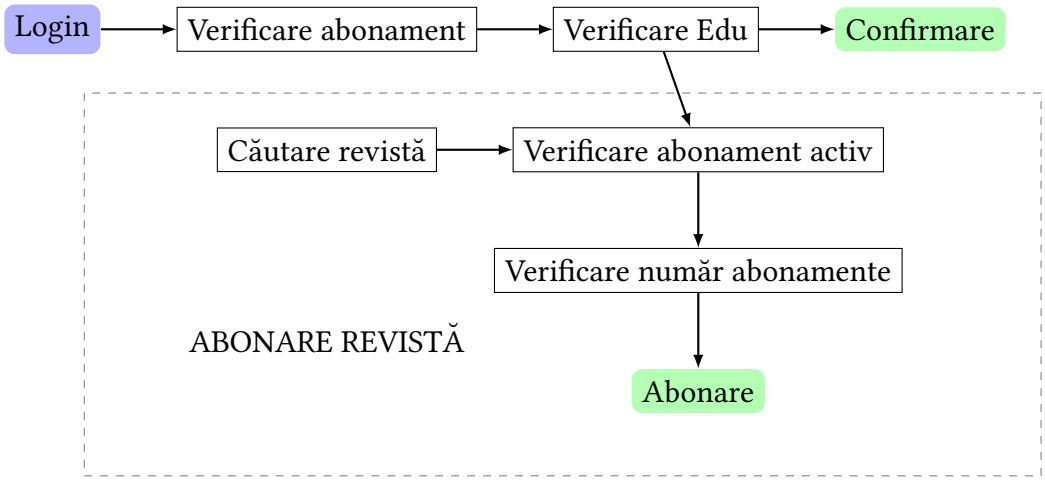

Figura 1.1: Descompunerea funcțională a procesului de abonare la revistă

Similar, procesul (P9) de verificare a statutului Edu se poate reprezenta ca în figura 1.2.

Și un ultim exemplu pe care îl prezentăm este acela al procesului (P16), de scriere a feedback-ului pentru un bibliotecar. Descompunerea funcțională este prezentată în figura 1.3.

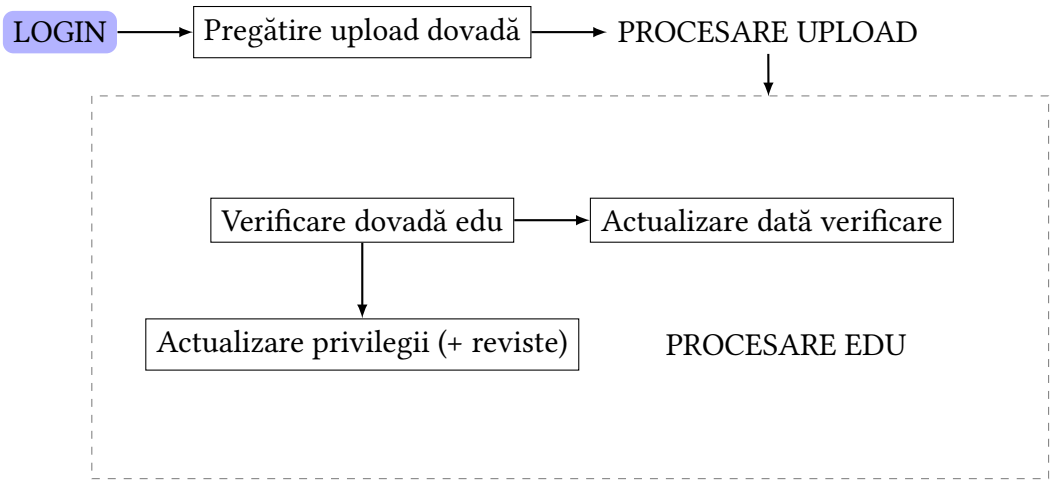

Figura 1.2: Descompunerea funcțională a procesului de verificare Edu

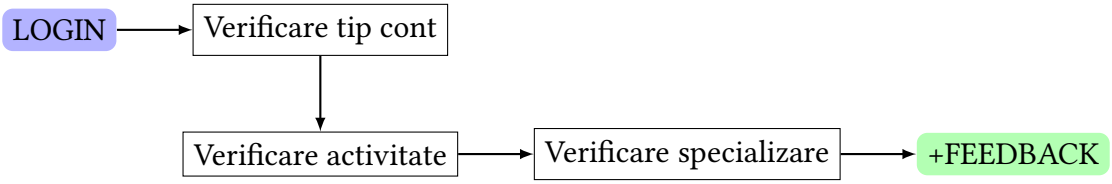

Figura 1.3: Descompunerea funcțională a procesului de adăugare feedback

1.3 Matricea proces-utilizator

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
Administrator	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Edu	X	X						X	X	X	X			X	X	X
NEdu	X	X						X	X	X				X	X	X
Bibliotecar	X	X	X	X	X				X	X	X	X	X	X		
Public	X									X				X		

1.4 Entitățile și modelarea datelor

Biblioteca este un centru de împrumut care permite și abonamentul la reviste pentru cei din mediul academic.

În bibliotecă se pot adăuga *cărți* și *reviste* de diverse specializări, în funcție de contractele cu furnizorii. Fiecare carte și revistă aparțin unei singure specializări.

Personalul Edu se poate abona și la reviste de specialitate, în baza unei verificări actualizate. Ceilalți cititori cu abonament pot deveni Edu printr-o verificare. Publicul larg poate doar să vadă stocul de cărți, general sau pe specializări și să adauge cărțile dorite în wishlist. Toți utilizatorii se pot abona la newsletter pentru a afla cînd se actualizează stocul.

De asemenea, bibliotecarii sînt asociați specializărilor, fiind responsabil de publicațiile dintr-o anumită specializare.

Cititorii abonați (`EDU` sau `NEDU`) pot lăsa feedback bibliotecarilor cu care au lucrat anterior într-o anumită specializare.

Diagrama ER se poate prezenta ca în figura 1.4.

De menționat că toate relațiile M:M ($\infty : \infty$) vor fi rezolvate cu *tabelele asociative* `BIB_SPEC`, `BIB_CIT` și `PUB_CARTE`, listate la §1.5.

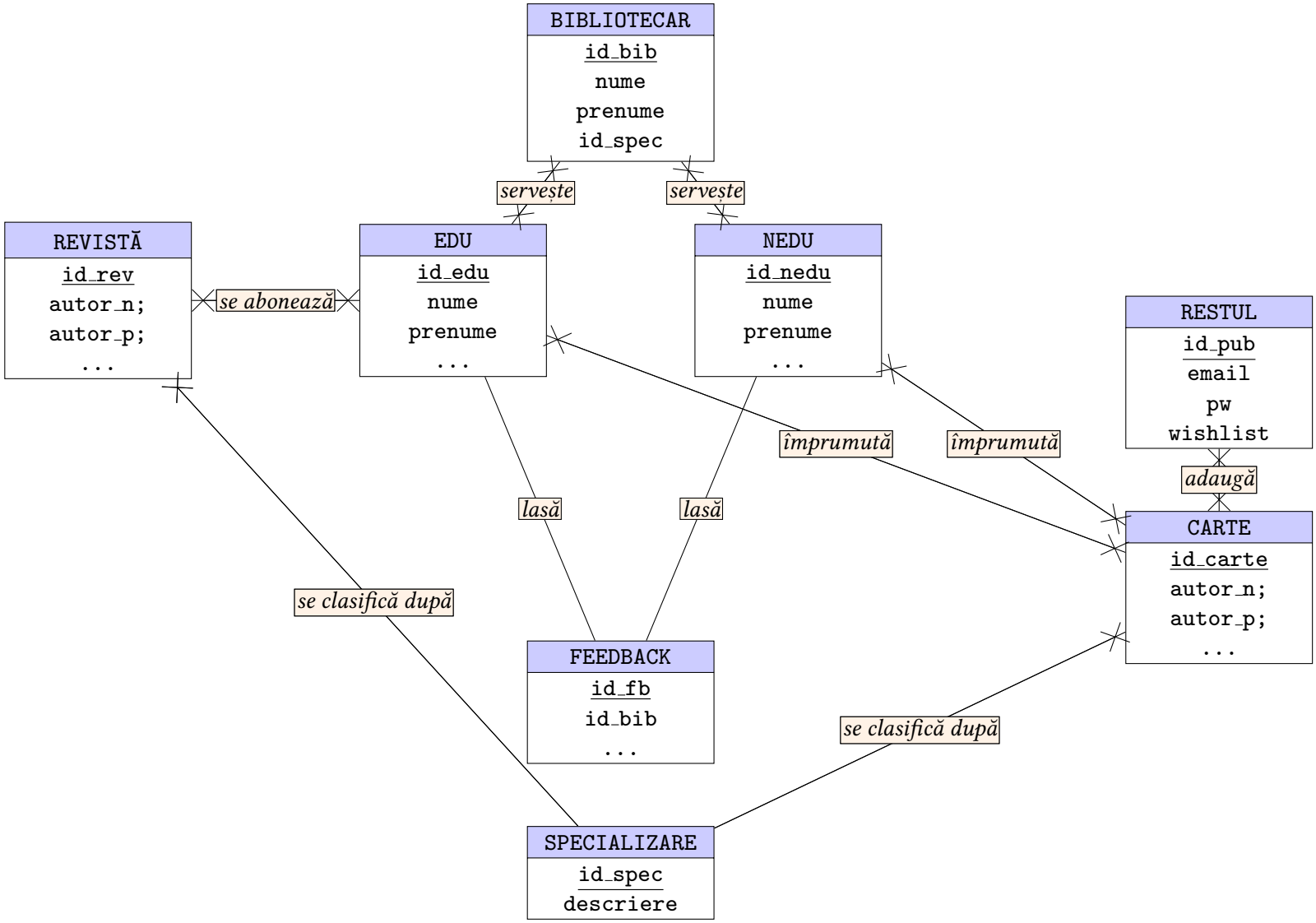

Figura 1.4: Diagrama ER

1.5 Schemele relaționale

```
BIBLIOTECAR     (id_bib#, nume, prenume, id_spec);
NEDU            (id_cit#, nume, prenume, email, newsletter, abonament_activ,
                 id_carte, id_spec);
EDU             (id_edu#, nume, prenume, email, newsletter, abonament_activ,
                 id_carte, id_rev, id_spec);
RESTUL          (id_pub#, email, pw, wishlist, newsletter);
CARTE           (id_carte#, autor_n, autor_p, titlu, an, id_spec, stoc);
REVISTA         (id_rev#, autor_n, autor_p, titlu, numar, id_spec, stoc);
SPECIALIZARE    (id_spec#, descriere);
FEEDBACK        (id_fb#, id_bib, id_cit, id_edu, id_spec, rating,
                 continut, datafb);
BIB_SPEC        (id_bib#, id_spec#);
BIB_CIT         (id_bib#, id_cit#);
PUB_CARTE       (id_pub#, id_carte#);
```

cum țin număr de împrumuturi & termen?

1.6 Matricea entitate-proces

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
BIBLIOTECAR	S	S	IUD	IUD	IUD		U	IUD	U	S	S	IUD	IUD	U	U	
NEDU	S	S							IUD	IUD	S				U	
EDU	S	S							U	IUD	S	S		U	U	
RESTUL	S					IUD						S		IUD		
CARTE	S		IUD								S				U	
REVISTA												S	I			
SPECIALIZARE											S	S				
FEEDBACK																I

Legenda: I = Insert, U = update, D = delete, S = select.

1.7 Matricea entitate-utilizator

	Edu	NEdu	Bibliotecar	Admin	Restul
BIBLIOTECAR	S	S	S	S, I, U, D	
NEDU		S, I, U, D	S, I, U, D		
EDU		S, I, U, D	S, I, U, D		
RESTUL					
CARTE	S	S	S, I, U, D	S, I, U, D	S
REVISTA	S		S, I, U, D	S, I, U, D	S
SPECIALIZARE	S	S	S	S, I, U, D	S
FEEDBACK	I	I	S	S,I,U,D	

Legenda: I = Insert, U = update, D = delete, S = select.

1.8 Utilizatori și conturi

Conturile utilizatorilor vor fi definite individual pentru utilizatorii identificați de bibliotecă.

În această etapă a dezvoltării aplicației, vor fi disponibile maximum 10000 de conturi Edu, 10000 de conturi de cititori (înregistrați), 10 conturi de bibliotecar, 1000 conturi de utilizator neabonat (RESTUL).

SECȚIUNEA 2

IMPLEMENTARE ÎN POSTGRESQL

2.1 Instalare și configurare inițială

Pentru implementare, vom folosi sistemul de baze de date PostgreSQL, disponibil gratuit și open source.

Implementarea se va face pe laptopul personal, folosind:

- OS: Manjaro Linux i3;
- Emacs 26 pentru editare text și comenzi shell;
- `st` pentru comenzi avansate de terminal, dacă este necesar.

Programul se instalează folosind managerul de pachete din Manjaro, cu comanda:

```
$ sudo pacman -S postgresql
```

După instalare, putem consulta consulta detaliile de pe pagina ArchWiki.

Pe scurt, PostgreSQL creează automat un utilizator cu numele `postgres`, care este proprietarul implicit al bazelor de date. Astfel, pentru a lansa aplicația și a face modificări, trebuie să ne identificăm ca utilizatorul `postgres`, cu una dintre comenzile:

```
$ sudo -iu postgres
$ su -l postgres
```

Inițializarea cluster-ului de baze de date, unde se vor crea bazele și tabelele se face în calitatea de utilizator `postgres`, de exemplu, în locația implicită, cu comanda:

```
[postgres]$ initdb -D /var/lib/postgres/data

# pentru a forța utilizarea limbii engleze și a codării UTF-8, folosim comanda
[postgres]$ initdb --locale=en_US.UTF-8 -E UTF8 -D /var/lib/postgres/data

# ... output ...
# finalizat cu ... ok
```

Folosind `systemd`, trebuie să activăm și să pornim daemon-ul `postgresql`, cu comenzile:

```
$ sudo systemctl enable postgresql
$ sudo systemctl start postgresql
```

Pentru început, creăm un utilizator nou, căruia îi putem da ce atribuții dorim, precum și o bază de date pe care utilizatorul respectiv să o poată accesa (sau administra, în funcție de rolul dat):

```
# devenim utilizatorul postgres mai întîi
$ sudo -iu postgres

# creăm utilizatorul (e.g. theUser) cu dialog pas cu pas, pentru a alege rolul
[postgres]$ createuser --interactive

# creăm o bază de date pentru el (e.g. theDatabase)
[postgres]$ createdb -O theUser theDatabase

# dacă theUser nu are rol de creare, putem crea cu postgres pentru el
[postgres]$ createdb -U postgres -O theUser theDatabase
```

Acum putem porni subshell-ul `psql` și să ne conectăm la baza de date creată mai sus:

```
$ sudo -iu postgres
[postgres]$ psql -d theDatabase

# cîteva dintre meta-comenzile pentru subshell-ul psql:
=> \help                    # accesează help
=> \c <database>            # conectează-te la baza de date <database>
=> \du                      # afișează utilizatorii și permisiunile
=> \dt                      # afișează tabelele și permisiunile
=> \q                       # închide subshell-ul
=> \?                       # afișează toate meta-comenzile
```

2.2 Crearea tabelelor

În primă fază, devenim utilizatorul `postgres` și accesăm baza de date creată:

```
[postgres]$ psql biblioteca
```

Putem încărca un script `psql` din fișierul `tabele.sql`, de exemplu după ce ne-am asigurat că fișierul `tabele.sql` are drepturi corespunzătoare, cu meta-comanda:

```
biblioteca# \i /calea/catre/script/tabele.sql
```

Apoi creăm tabelele corespunzătoare schemei din §1.5 folosind scriptul:

```
create table bibliotecar (
   id_bib bigserial not null primary key,
   -- bigserial = crește singur
   nume varchar(20) not null,
   prenume varchar(20) not null,
   id_spec varchar(20) not null
)

create table edu (
   id_edu bigserial not null primary key,
   nume varchar(20) not null,
   prenume varchar(20) not null,
   email varchar(20),
   newsletter boolean,
   ab_activ boolean not null,
   foreign key (id_carte_edu) references carte(id_carte),
   foreign key (id_rev_edu) references revista(id_rev),
   foreign key (id_spec_edu) references specializare(id_spec)
)

create table nedu (
   id_cit bigserial not null primary key,
   nume varchar(20) not null,
   prenume varchar(20) not null,
   email varchar(20),
   newsletter boolean,
   ab_activ boolean not null,
   foreign key (id_carte_nedu) references carte(id_carte),
   foreign key (id_spec_nedu) references specializare(id_spec)
)

create table restul (
   id_pub bigserial not null primary key,
```

```
   email varchar(20),
   pw varchar(20),
   wishlist text[],-- vector de cuvinte
   newsletter boolean
)

create table carte (
   id_carte bigserial not null primary key,
   autor_n varchar(20) not null,
   autor_p varchar(20) not null,
   titlu varchar(30) not null,
   an smallserial,-- 1 -> 32767
   stoc smallint,-- -32768 ->
   foreign key (id_spec_carte) references specializare(id_spec)
)

create table revista (
   id_rev bigserial not null primary key,
   autor_n varchar(20) not null,
   autor_p varchar(20) not null,
   titlu varchar(30) not null,
   numar smallserial,
   stoc smallint,
   foreign key (id_spec_rev) references specializare(id_spec)
)

create table specializare (
   id_spec bigserial not null primary key,
   descriere text
)

create table feedback (
   id_fb bigserial not null primary key,
   rating smallserial,
   continut text,
   datafb date,
   foreign key (id_bib_fb) references bibliotecar(id_bib),
   foreign key (id_cit_fb) references nedu(id_cit),
   foreign key (id_edu_fb) references edu(id_edu),
   foreign key (id_spec_fb) references specializare(id_spec),
```

```
)

create table bib_spec (
   foreign key (id_bib_bs) references bibliotecar(id_bib) on delete restrict,
   -- nu se șterg tabelele cu referințe pînă ce toate referințele s-au șters
   foreign key (id_spec_bs) references specializare(id_spec) on delete restrict,
)

create table bib_cit (
   foreign key (id_bib_bc) references bibliotecar(id_bib),
   foreign key (id_cit_bc) references nedu(id_cit)
)

create table pub_carte (
   foreign key (id_pub_pc) references restul(id_pub),
   foreign key (id_carte_pc) references carte(id_carte)
)
```

Am adăugat deja constrîngeri în scriptul de mai sus, dar cheile străine pot fi definite și separat, ca în exemplul de mai jos:

```
-- creăm coloana care va deveni cheie străină:
biblioteca# alter table bib_cit
biblioteca# add column id_bib_bc bigint;
ALTER TABLE
biblioteca# alter table bib_cit
biblioteca# add constraint fk_id_bib_bc foreign key (id_bib_bc)
biblioteca# references bibliotecar(id_bib);
ALTER TABLE
```

În felul acesta, legăm coloana `ib_bib_bc` de coloana `bibliotecar(id_bib)`, iar constrîngerea o numim `fk_id_bib_bc`.

După aceea, populăm fiecare dintre tabele cu înregistrări. De exemplu, putem începe cu tabelul care nu conține referințe, `specializare`:

```
biblioteca# insert into specializare values (1, 'matematica'),
                                            (DEFAULT, 'literatura'),
                                            (DEFAULT, 'politica');
INSERT 0 1
```

Valoarea `DEFAULT` continuă automat numărătoarea, deoarece cîmpul respectiv este `id_spec`, declarat cu tipul `bigserial`.

2.3 Aspecte de securitate