Universitatea Tehnică „Georghe Asachi” din Iaşi
Facultatea de Automatică şi Calculatoare
Specializarea Calculatoare şi Tehnologia Informaţiei
Disciplina Baze de Date Proiect

Gestiunea activității unei biblioteci

Profesor Coordonator
Avram Sorin

Student
Hușman Carla-Gabriela
Grupa 1311A

Iași, 2022

Titlu proiect: Gestiunea activității unei biblioteci

Proiectul va viza analiza problemei de gestionare a unei biblioteci, proiectarea și implementarea unei baze de date care să permită modelarea fluxului de evenimente care trec printr-o bibliotecă.

Descrierea cerințelor și modul de organizare al proiectului

Pentru a conduce o bibliotecă în mod armonios și pentru a avea dosarele scrise într-un singur loc, este necesară o bază de date care să transfere toate aceste informații într-un mediu de încredere, fără a avea grija pierderii lor.

Activitatea de gestiune a unei biblioteci implică ținerea evidenței clienților, a datelor lor personale, a cărților care sunt furnizate de anumite edituri care doresc să devină colaboratori. Cărțile pot fi date spre închiriere doar dacă sunt pe stoc, deci aplicația facilitează timpul de căutare al disponibilității cărții, nemaifiind nevoie de căutarea fizică a acesteia printre rafturile bibliotecii.

În final, închirierile (tranzacții) se vor realiza cu posibilitatea deplasării din incinta bibliotecii cu cartea dorită. Toate acestea vor putea fi realizate cu succes doar dacă respectivul produs mai este în stoc. Cartea va trebui să fie returnată după 7 zile, dar există posibilitatea prelungirii termenului de închiriere cu câte zile are nevoie clientul. Totodată, este permis ca inchirierea să se încheie și mai devreme de 7 zile. Odată cu aducerea cărții la ghișeu, se va ține cont de starea în care a fost returnată.

Informațiile de care avem nevoie sunt cele legate de:

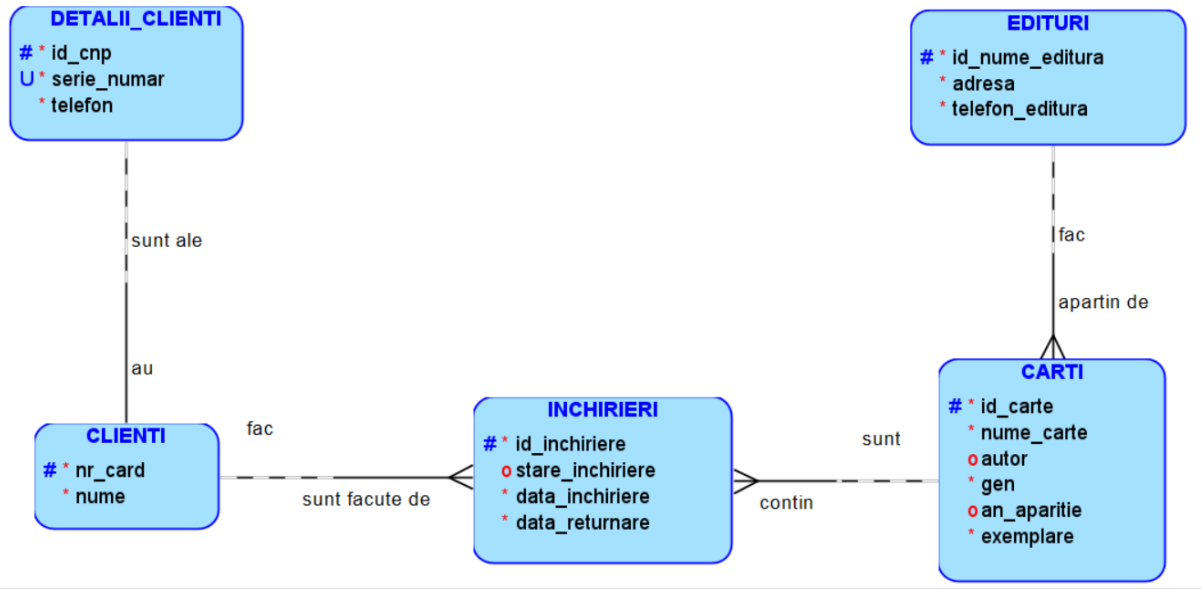
- **Detaliile clienților**: ne interesează anumite informații unice fiecărui client pentru a-i putea identifica, deosebi și contacta în caz de urgențe, cum ar fi cnp, seria, numărul de la buletin și numărul de telefon
- **Clienții**: avem nevoie să reținem o listă cu clienții bibliotecii, fiecare având câte un card unic de identificare care reține informațiile sale
- **Editurile**: sunt un fel de furnizori, care oferă bibliotecii cărți spre închiriere. Este necesară evidența acestora pentru a îi contacta pentru viitoare livrări de cărți sau pentru eventualele probleme ale coletelor
- **Cărțile**: entitatea în care vom reține toate datele legate de fiecare carte pe care biblioteca o posedă, precum titlul, autorul, anul apariției, genul, numărul de exemplare, editura
- **Închirierile**: este cea mai importantă entitate, cât și nucleul bibliotecii, în care se ține evidența tranzacțiilor care se realizează atunci când o carte este închiriată

Descrierea funțională a aplicației:

- Evidența operațiunilor referitoare la închirierea cărților
- Evidența clienților
- Evidența cărților

Descrierea detaliată a modelului logic, a entităților și a relațiilor dintre tabele:

1. Diagrama modelului logic:



2. Entitățile și constrângerile folosite:

   ❖ DETALII_CLIENȚI – entitate folosită în gestionarea informațiilor fiecărui client (toate atributele sunt obligatorii)

   1. id_cnp – NUMBER(13)
      - id_cnp_pk (Primary Key)
      - id_cnp_ck: verificarea ca cnpul să nu conțină litere sau alte caractere, dar să fie alcătuit din 13 cifre (cu regex)

   2. serie_număr – VARCHAR2(9) NOT NULL
      - serie_număr_ck: verificarea ca să fie alcătuit din 2 litere sau litera B (București), un spațiu și 6 cifre (cu regex)
      - serie_număr_uk: impun ca o combinație dintre serie și număr să fie unică

   3. telefon – VARCHAR2(10) NOT NULL
      - telefon_ck: verificarea ca numărul de telefon să fie alcătuit din 10 cifre, fără litere sau alte caractere și să înceapă cu 07 (cu regex)

   4. *nr_inchirieri – am avut înainte acest atribut, dar am renunțat la el deoarece era inutil, putând să aflu numărul de închirieri dând un Count în tabela ÎNCHIRIERI*

   5. *școlar – am avut acest atribut înainte și lucrând pe baza de date, inserând și modificând l-am considerat redundant deoarece nu îl foloseam nicăieri, nu făceam legătura dintre el și atributul pret_redus din CĂRȚI. Am ales să renunț la atributele care poluau entitatea*

- CLIENȚI – entitate folosită în gestionarea clienților magazinului și a cardurilor acestora (toate atributele sunt obligatorii)
  - nr_card – NUMBER (7)
    - nr_card_pk (Primary Key)
  - id_cnp – NUMERIC(13) NOT NULL
    - id_cnp_fk (Foreign Key): asigură legătura cu entitatea DETALII_CLIENȚI
  - nume – VARCHAR2(20) NOT NULL
    - nume_ck: verificarea ca numele să aibă numai caractere, să nu aibă cifre și să permită spațiere între numele de familie și prenume (cu regex)
- EDITURI – entitate folosită pentru gestionarea furnizorilor care aprovizionează biblioteca (toate câmpurile sunt obligatorii)
  - id_nume_editură – VARCHAR2(20)
    - id_nume_editură_pk (Primary Key)
    - id_nume_editură_ck: verificarea ca numele editurii să conțină doar litere, fără cifre sau alte caractere (cu regex)
  - adresă – VARCHAR2(40) NOT NULL
    - adresă_ck: verificarea ca adresa să permită spațierea, punctul, virgula, cifrele și literele (cu regex)
  - telefon_editură - VARCHAR2(10) NOT NULL
    - telefon_editură_ck: verificarea ca numărul de telefon să fie alcătuit din 10 cifre, fără litere sau alte caractere și să înceapă cu 07 (cu regex)
- CĂRȚI – entitate ce conține informațiile cărților din bibliotecă
  - id_carte – NUMBER(3)
    - id_carte_pk (Primary Key)
  - id_nume_editura – VARCHAR2(20) NOT NULL
    - id_nume_editura_fk (Foreign Key): asigură legătura cu entitatea EDITURI
  - nume_carte – VARCHAR2(35) NOT NULL
    - nume_carte_ck: verificarea ca titlul cărții să poată conține litere, cifre, spații, cât și punctul (cu regex)
  - autor – VARCHAR2(30)
    - autor_ck: verificarea ca numele cărții să conțină litere, spații și caracterul punct (cu regex)
    - nu este un câmp obligatoriu deoarece nu este cunoscut întotdeauna autorul unei cărți

- gen – VARCHAR2(20) NOT NULL
  - gen_ck: verificarea ca genul să conțină doar litere (cu regex)

- an_apariție – NUMBER (4)
  - an_apariție_ck: verificarea ca anul apariției să fie cuprins între 1000 și 2024, deoarece o carte nu poate fi publicată în viitor
  - nu este un câmp obligatoriu deoarece nu este cunoscut anul de apariție al oricărei cărți

- exemplare – NUMBER (2) NOT NULL
  - exemplare_ck: verificarea ca să fie mai mare sau egal cu 0. Zero însemnând că nu mai am pe stoc, deci nu mai pot fi închiriate

- *preț și preț_redus – am ales să renunț la aceste atribute deoarece nu aveau niciun rol. Nu mă foloseam de ele și nici nu aveam o entitate care să stocheze venitul pe care îl faceam în urma închirierilor. Atributul pret_redus ar fi fost legat de atributul școlar din tabela DETALII_CLIENȚI pentru a oferi studenților/elevilor o reducere. Nefolosind niciuna dintre aceste facilități, le-am eliminat*

❖ ÎNCHIRIERI – entitate folosită pentru gestionarea închirierilor realizate de către clienți

- id_închiriere – NUMBER(2)
  - id_inchiriere_pk (Primary Key)

- id_carte – NUMBER(5) NOT NULL
  - id_carte_fk (Foreign Key): asigură legătura cu entitatea CĂRȚI

- nr_card – NUMBER(7)
  - nr_card_fk (Foreign Key): asigură legătura cu entitatea CLIENȚI

- stare_închiriere – VARCHAR2(8)
  - stare_inchiriere_ck: verificare ca starea să fie una din următoarea listă: bună, uzată, pierdută
  - aceasta va fi NULL pe parcursul închirierii, iar la returnare va primi una dintre valorile menționate anterior (la returnare este posibilă schimbarea dății de returnare)

- dată_închiriere – DATE NOT NULL
  - poate fi data de astăzi sau o dată din trecut, dar niciodată una din viitor (validare în interfață)

- dată_returnare – DATE NOT NULL
  - poate fi orice dată, numai nu înaintea celei de închiriere (validare în interfață)

3. Tipurile de relații:

- 1:1
  - Între CLIENȚI și DETALII_CLIENȚI - deoarece fiecare intrare în entitatea DETALII_CLIENȚI va avea un corespondent unic în entitatea CLIENȚI.

- 1:n
  - Între CLIENȚI și ÎNCHIRIERI - deoarece entitatea ÎNCHIRIERI conține istoricul tuturor închirierilor, existând posibilitatea ca un client să facă mai multe închirieri, dar o închiriere nu poate fi făcută de mai multi clienți
  - Între CĂRȚI și ÎNCHIRIERI - deoarece cărțile pot fi închiriate de mai multe ori de către clienți, dar unei închirieri îi va corespunde o singură carte (aceeași carte, ca obiect palpabil, nu poate fi în același timp la doi clienți).
  - Între EDITURI și CĂRȚI – deoarece o editură poate furniza mai multe cărți bibliotecii, dar o carte nu poate avea mai multe edituri

Normalizarea bazei de date

Normalizarea a fost folosită la nivelul entităților CLIENȚI, CĂRȚI și ÎNCHIRIERI prin folosirea foreign key-urilor pentru a le descompune în entități mai mici care stochează aceleasi date ca și entitatea inițială astfel încât să fie eliminate redundanța în date și anomaliile la actualizare.

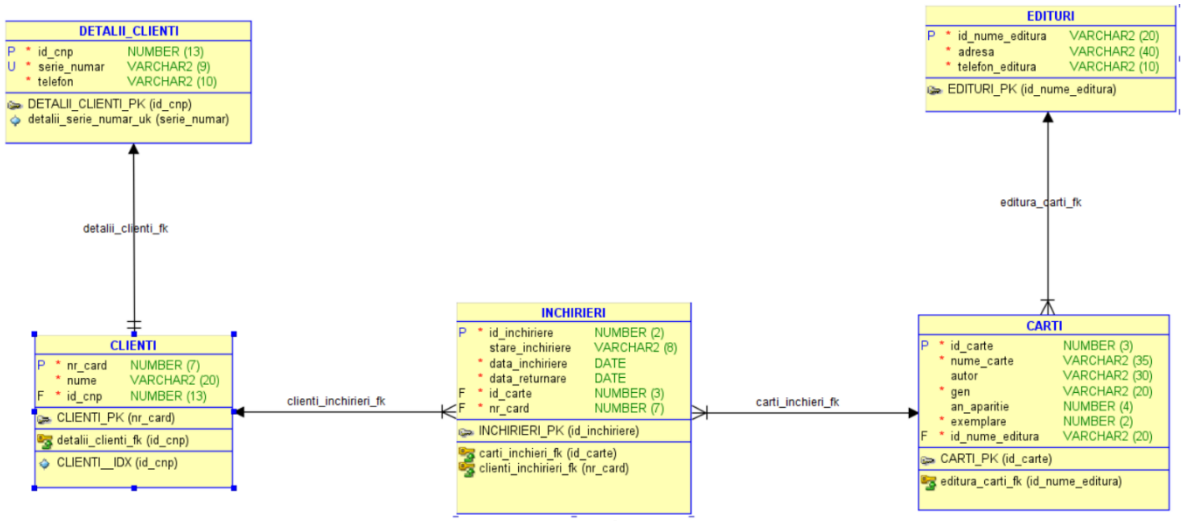
Normalizarea entității:

- CĂRȚI
  - id_nume_editură – numele editurii ar fi fost speficicat de N ori, această repetiție fiind redundantă. S-a folosit astfel o tabelă EDITURI care să stocheze fiecare editură împreună cu informațiile sale aferente, fără a popula excesiv entitatea CĂRȚI

- ÎNCHIRIERI
  - id_carte, nr_card – dacă toate acestea ar fi fost reunite în entitatea ÎNCHIRIERI, ar fi fost probleme la introducerea de informații deoarece, în cazul în care un client ar fi dorit să-și facă un card, dar nu să închirieze în acel moment o carte, restul câmpurilor care ar fi fost obligatorii din ÎNCHIRIERI ar fi rămas necompletate => operațiune eșuată => ANOMALIE la inserare

- CLIENȚI
  - id_cnp – idem ca la cele explicate anterior

Descrierea detaliată a modelului relațional:



Pentru cheile primare CĂRȚI.id_carte și ÎNCHIRIERI.id_inchiriere, s-a ales folosirea funcției de autoincrement pusă la dispoziție de data modeler, care are la bază IDENTITY, pentru a permite migrarea mai ușoară în, și din alte SGBDuri (s-a folosit Oracle 12c). La restul primary key-urilor nu s-a folosit autoincrementul deoarece este necesară introducerea la mână a acestora, de ex. cnpul, numărul cardului sau numele editurii.

Funcțiile folosite:

- DISPONIBILITATE: primește ca parametru id-ul unei cărți care se dorește a fi închiriată și returnează id-ul acesteia dacă mai este în „stoc” (adică dacă numărul de exemplare este mai mare decât numărul de bucăți închiriate din acel exemplar). În caz contrar, va returna NULL, pentru ca închirierea să nu mai aibă loc.

Descrierea detaliată a interfeței:

Aplicația are ca utilizatori angajații bibliotecii. Funcționalitățile pe care se bazează aplicația sunt vizualizarea, modificarea, adăugarea, ștergerea:

- clienților
- cărților
- editurilor
- închirierilor

Conectarea la baza de date

Se face prin intermediul modulului cx_Oracle, folosind instrucțiunea cx_Oracle.connect (username, parolă, nume_domeniu: port/serviciu).

Tehnologiile utilizate

- **Back-end**: Python, SQL, Django
- **Front-end**: HTML, CSS

Aplicația poate fi folosită doar din modul de administrator, adică de către un angajat al bibliotecii care are dreptul să realizeze orice schimbare și să realizeze tranzacții.

În continuare, pentru vizualizarea tuturor opțiunilor descrise, se va atașa print screenuri diferite la fiecare opțiune a meniului.

Prezentarea interfețelor și a modului de utilizare

La pornirea aplicației, administratorul/angajatul va vedea meniul aplicației de unde poate alege dintre 4 opțiuni:

1. CLIENȚI

   - Unde se va deschide pagina destinată afișării tuturor clienților
   - Dacă se apasă pe numele clientului va fi posibilă vizualizarea detaliilor individului
   - Modificarea (buton de Editare în dreptul fiecărui client) se va realiza doar la nivelul atributelor *nume* și *telefon* din motive de securitate. Cnpul, seria, numărul și numărul cardului fiind unice fiecărui client, aceastea nu se vor schimba
   - Clientul poate fi șters (buton de Ștergere în dreptul fiecărui client) din baza de date numai dacă acesta nu a făcut niciodată o închiriere
   - Adăugarea clientului (buton) se va realiza atât la nivelul entității CLIENȚI, cât și la nivelul entității DETALII_CLIENȚI prin informațiile inserate în pagina de adăugare

2. CĂRȚI

   - Pagina destinată vizualizării tuturor cărților și a informațiilor despre acestea. Prezintă aceleași facilități ca mai sus, adăugare, editare, ștergere (doar dacă respectiva carte nu este data spre închiriere)

- Editarea se poate realiza la nivelul tuturor atributelor. Am lăsat această libertate deoarece, în cazul introducerii greșite a unei informații, angajatul să poată schimba imediat

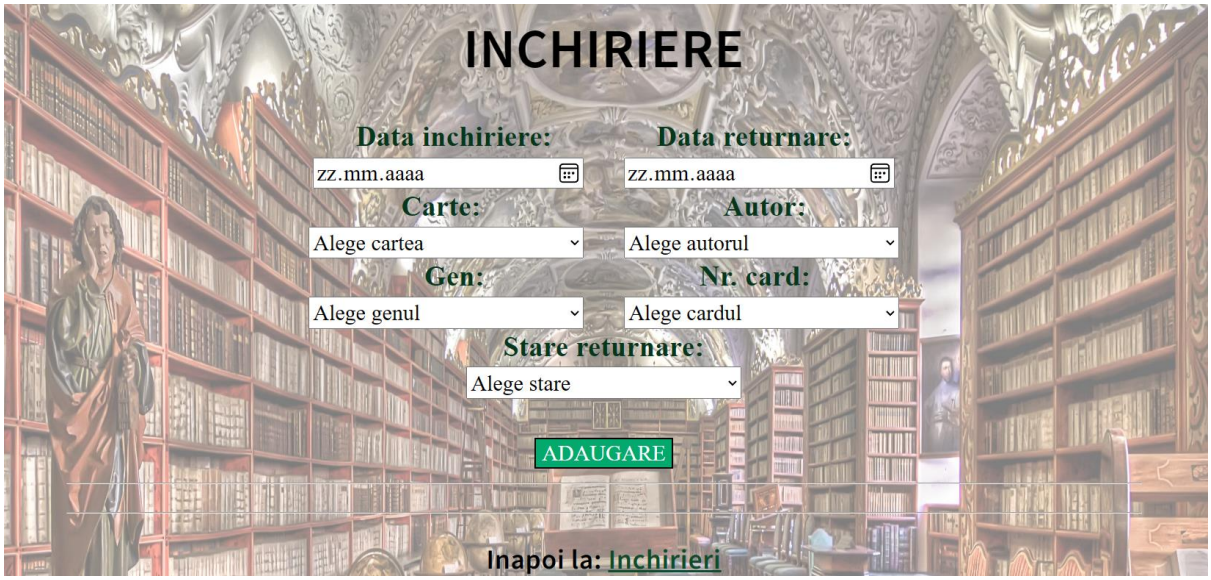
3. EDITURI

- Pagina destinată vizualizării furnizorilor și a informațiilor despre aceștia. Prezintă aceleași facilități ca mai sus, adăugare, editare, stergere
- Editarea se poate realiza doar la nivelul atributelor adresă și telefon, numele editurii neputând fi schimbat din motive de securitate. O editură își poate schimba adresa sau numărul de telefon, dar nu și numele. Chiar dacă acest lucru ar fi posibil, cărțile de la vechiul nume sunt încă în bibliotecă, deci este importantă păstrarea denumirii editurii
- O editură poate fi ștearsă doar dacă nu sunt găsite cărți în baza de date de la aceasta

DENUMIRE	ADRESA	TELEFON	EDITARE/STERGERE
Globo	Bulevardul Poitiers 6, Iasi	0761111111	editare stergere
Libra	Bulevardul Nicolae Balcescu 20, Cluj	0761111112	editare stergere
Arthur	Str. Cotroceni nr. 26, Bucuresti	0761111113	editare stergere
Paralela	Strada Sfantul Lazar nr. 27, Iasi	0761111114	editare stergere
Litera	Bulevardul Carol I 3, Iasi	0761111115	editare stergere
Bloomsbury	Bulevardul Pantelimon 54, Iasi	0761111225	editare stergere

4. ÎNCHIRIERI

- Permite vizualizarea tuturor închirierilor
- Editarea unei închirieri este posibilă doar la nivelul câmpurilor de stare și data_returnare. Starea se completează la restituire, iar data de returnare poate fi schimbată, cartea fiind înapoiată ori mai devreme de termenul trecut inițial, ori mai târziu (posibilitate de prelungire)
- Se poate realiza ștergerea oricărei închirieri
- Tranzacțiile se pot realiza prin introducerea de informații în toate câmpurile (opțional este doar stare returnare). În cazul unei defecțiuni a sistemului de o săptămână, angajatul își va trece pe foaie toate detaliile închirierii. Este posibilă returnarea cărții în acest interval. Odată ce sistemul își revine, angajatul va introduce toate câmpurile



Notă: La inserarea în oricare entitate, în cazul în care s-a omis introducerea de informații în câmpurile destinate atributelor obligatorii din entități, se va afișa un mesaj care indică acest lucru. Insertul nu se va realiza până când nu sunt completate aceste câmpuri. În caz de introducere eronată, după apăsarea butonului de Adăugare, va apărea un mesaj de eroare specific câmpului. Aceleași precizări sunt și pentru editarea informațiilor. În cazul ștergerilor, dacă ștergerea nu reușește, se va afișa un mesaj de eroare.

Unele informații vor fi alese dintr-o listă. De exemplu: numele editurii la editare/inserare carte, stare, carte, card, autor, gen la nivelul tranzacției. Dățile vor fi selectate de pe un calendar.

Exemple de cod și instrucțiuni sql folosite

S-a folosit un simplu select pentru a extrage toate informațiile din clienți și pentru a le pasa htmlului.

```
def clienti(request):
    template = loader.get_template('clienti.html')

    clienti = []

    cur = connection.cursor()
    cur.execute('select * from clienti')
    for result in cur:
        client = {}
        client['nr_card'] = result[0]
        client['nume'] = result[1]
        client['id_cnp'] = result[2]
        clienti.append(client)

    cur.close()

    context = {
        'client': clienti,
    }

    return HttpResponse(template.render(context, request))
```

S-a folosit un join pentru a extrage detaliile unui anumit client dat ca parametru prin id_cnp.

```
def detalii(request, id_cnp):
    template = loader.get_template('detalii.html')

    cur1 = connection.cursor()
    sql = "select nume, serie_numar, telefon " \
          "from detalii_clienti, clienti " \
          "where detalii_clienti.id_cnp=clienti.id_cnp " \
          "and clienti.id_cnp=" + str(id_cnp)
    cur1.execute(sql)

    detalii = []
    for result in cur1:
        detaliu = {}
        detaliu['nume'] = result[0]
        detaliu['serie_numar'] = result[1]
        detaliu['telefon'] = result[2]
        detaliu['id_cnp'] = id_cnp
        detalii.append(detaliu)

    context = {
        'detalii': detalii,
    }
    cur1.close()
```

Comenzi sql executate pentru a se putea realiza ștergerea unui client.

```
def stergereClient(request, nr_card):
    template = loader.get_template('stergereClient.html')

    cur = connection.cursor()
    try:
        cur.execute('select id_cnp from clienti where nr_card=' + str(nr_card))
        for result in cur:
            cnp = result[0]
        cur.execute('delete from clienti where nr_card=' + str(nr_card))
        cur.execute('delete from detalii_clienti where id_cnp=' + str(cnp))
        cur.execute('commit')
        mesaj = "Operatiune efectuata cu success!"
    except:
        mesaj = "ACTIUNE INVALIDA! Stergerea nu a reusit." \
                " Clientul pe care doriti sa-l stergeti este in baza de " \
                "date a inchirierilor."

    cur.close()

    context = {
        'mesaj': mesaj,
    }

    return HttpResponse(template.render(context, request))
```

Update-ul se va face individual pentru fiecare atribut doar dacă a fost introdusă o nouă valoare. În cazul introducerii unei valori invalide, se va afișa un mesaj de eroare.

```
if adresa != editura['adresa']:
    var = "'" + adresa + "'"
    var2 = "'" + id_editura + "'"
    try:
        cur.execute(
            'update edituri set adresa=' + var +
            ' where id_nume_editura=' + var2)
        cur.execute('commit')
        mesaj = 'Operatiune completa!'
    except:
        mesaj = 'Editarea nu a reusit. Adresa invalida!'
        flag = 0

if telefon != editura['telefon']:
    var = "'" + telefon + "'"
    var2 = "'" + id_editura + "'"
    try:
        cur.execute(
            'update edituri set telefon_editura=' + var +
            ' where id_nume_editura=' + var2)
        cur.execute('commit')
        mesaj = 'Operatiune completa!'
    except:
        mesaj = 'Editarea nu a reusit. Telefonul este introdus gresit!'
        flag = 0
```

Înaintea oricărei inserări, s-au validat datele de intrare prin verificarea regexurilor. S-au făcut verificări în cazul în care se introduce o cheie primară deja existentă. În cazul inserării foreign key-urilor sau a valorilor care se aleg dintr-o listă, nu pot exista erori deoarece respectiva lista constrânge angajatul la introducerea unor valori valide.

```
pat = re.compile(r"^(([A-Z]){2}|B)( )([0-9]){6}$")
if re.fullmatch(pat, serie):
    mesajSerie = ''
else:
    mesajSerie = 'Format invalid'
    flag = 0

cur.execute('select id_cnp from detalii_clienti')
UNIQUE = 1
for result in cur:
    var = str(result[0])
    if var == cnp:
        UNIQUE = 0

pat = re.compile(r"^[1-9]{1}[0-9]{12}$")
if UNIQUE == 0:
    mesajCnp = 'Ati introdus un CNP existent'
    flag = 0
elif re.fullmatch(pat, cnp):
    mesajCnp = ''
else:
    mesajCnp = 'Format invalid'
    flag = 0
```

```
pat = re.compile(r"^([0-9]){10}$")
if re.fullmatch(pat, telefon):
    mesajTelefon = ''
else:
    mesajTelefon = 'Introduceti doar 10 cifre'
    flag = 0
```

```
if flag == 1:
    mesaj = 'Inserare completa!'
    var1 = "'" + serie + "'"
    var2 = "'" + telefon + "'"
    cur.execute('INSERT INTO detalii_clienti '
                'VALUES (' + str(cnp) + ',' + var1 + ',' + var2 + ')')
    var1 = "'" + nume + "'"
    cur.execute('INSERT INTO clienti '
                'VALUES (' + str(card) + ',' + var1 + ',' + str(cnp) + ')')
    cur.execute('commit')
else:
    mesaj = 'Inserarea a esuat!'
```

S-au făcut validări și pentru datele calendaristice. Data_returnării nu poate fi mai mică decât data închirierii, iar data închirierii nu poate fi una din viitor (temp1 fiind dată_închiriere, iar temp2 fiind dată_returnare).

```
if temp1 > date.today():
    mesaj = mesaj + '\n' \
                    'Data de inchiriere nu poate fi din viitor.'
    flag = 0

if temp2 < temp1:
    mesaj = mesaj + '\n' \
                    'Data returnare nu poate fi inaintea celei de inchiriere.'
    flag = 0
```