UNIVERSITATEA TEHNICĂ "GH ASACHI" IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA BAZE DE DATE PROIECT

Gestiunea unui cinematograf

Coordonator,

Mironeanu Catalin

Student,

Matei Rares – 1306A

- **Cerintele si organizarea proiectului**

Baza de date este creata pentru a gestiona vanzarile biletelor care se fac intr-un cinematograf, stocarea filmelor si detaliilor acestora, a salilor, a clientilor, a programului unui film si a tipurilor de reducere de care pot beneficia clentii in urma validarii statutului printr-un act (carnet de student, carnet de pensionar, carnet de elev).

Cinematograful are un numar limitat de sali, in care se difuzeaza cate un film de-a lungul zilei, facandu-se o pauza de 40 de minute intre orele de incepere a filmelor. Numarul de sali si detaliile acestora cat si filmele si programul acestora este decis de conducerea cinematografului.

O data ce filmul se termina, inregistrarea acestuia de la ora respectiva este stearsa iar biletele corespunzatoare acelei combinatii de film si ora devin nule.

Biletele se vand in limita numarului de locuri libere din sala de la ora specificata, locul dorit fiind unicat.
Un client poate cumpara un numar nelimitat de bilete, pentru orice film de la orice ora inregistrata cat timp respecta conditiile anterioare.

Clientul va putea beneficia de o singura reducere de tip: student, elev, pensionar, copil (reducerea se aplica la toate biletele achizitionate de clientul respectiv).

Informatiile necesare sunt:

- Numele, durata, anul de lansare a unui film
- Numarul de randuri si de locuri a unei sali
- Statutul unui client, data, ora, locul si randul dorit

Aplicatia va avea urmatoarele functii:

- Stocarea datelor legate de film
- Afisarea filmelor disponibile in functie de gen, de la o anumita ora
- Biletele valide achizitionate de un client
- Afisarea clientilor care au bilete valide
- Numarul de locuri libere la un film de la o ora specificata

- **Tehnolgiile folosite pentru Front-end si Back-end**

  Pentru Front-end

  Am utilizat libraria cx_Oracle pentru a realiza conexiunea intefetei cu baza de date, libraria PyQt6 pentru GUI prin intermediul Qt Designer si limbajul de programare python pentru functionalitati.

  Pentru Back-end

  Am utilizat Oracle SQL Developer, Oracle SQL Developer Data Modeler pentru crearea tabelelor, realizarea modelului logic si cel relațional.

- **Structura proiectului, inter-relationarea entitatilor si normalizarea bazei de date**

  Entitatile si atributele folosite in aplicatie sunt:

  - FILM: detine id-ul, durata in minute, numele filmului, data lansarii filmului.
  - DETALII_FILM: contine id-ul filmului si detalii legate de acesta: nota, id-ul actorului principal, id-ul genului.
  - SALA: detine id-ul, numarul de locuri totale si de randuri ale unei sali.
  - BILET: fiecare bilet are id-ul sau, filmul, sala, pretul dupa reducere,id-ul reducerii, numarul locului si randul pe care se afla locul, data si ora flmului.
  - CLIENT: contine id-ul clientului, varsta si statutul sau.
  - TIP_BILET: detine id-ul, numele si pretul reducerii care se aplica la pretul biletului.
  - COMBINATIE: contine id-ul, id-ul filmului, id-ul salii, data si ora la care e programt filmul si numarul de locuri libere.
  - ACTOR: contine id-ul si numele actorului.
  - GEN_FILM: contine id_ul si numele genului

  In proiectarea acestei baze de date s-au identificat urmatoarele tipuri de relatii:

  - 1:1 (one-to-one)
  - 1:n (one-to-many)
  - n:n (many-to-many)

  Relatia 1:1 este realizata de catre tabelele film si detalii_film, deoarece un film are propriile detalii, desi, in acest caz unele particularitati nu pot fi identificate, deci e optional ca un film sa aiba detalii. Legatura dintre cele doua tabele este realizata prin cheia id_film.

  Relatiile 1:n sunt realizate de catre:

- BILET si COMBINATIE: O combinatie poate defini mai multe bilete(dar poate sa nu defineasca niciun bilet) iar un bilet trebuie sa aiba o combinatie . Legatura dintre cele doua tabele este realizata prin campu id_combinatie.
- BILET si TIP_BILET: Un bilet are un singur tip dar un tip de bilet se poate aplica la mai multe bilete, in acelasi timp un tip de bilet poate sa nu fie aplicat nici unui bilet. Legatura dintre cele doua tabele este realizata prin campul id_tip.
- CLIENT si BILET: Un bilet trebuie sa aiba un client iar un client poate cumpara unu sau mai multe bilet, totusi clientul poate exista si fara sa cumpere un bilet. Legatura dintre cele doua tabele este realizata prin campul id_client.
- ACTOR si DETALII_FILM: Un actor se poate incadra in una sau mai multe detalii de film dar o entitate detalii_film poate avea doar un actor.
- GEN_FILM si DETALII_FILM: un film poate fi de un singur gen dar mai multe filme pot fi de un anumit gen.

Relatia n:n este realizata printr-o tabela intermediara (COMBINATIE) deoarece un bilet are o sala si un film care pot fi alese in moduri diferite, la ore diferite, care formeaza 2 relatii de tip 1:n astfel:

- Film si Combinatie, deoarece un film poate face parte din una sau mai multe combinatii (filmul poate exista si fara sa fie pus intr-o combinatie) iar o combinatie trebuie sa aiba un film . Legatura dintre cele doua tabele este realizata prin campul id_film.
- Sala si Combinatie, deoarece o sala poate face parte din una sau mai multe combinatii (sala poate exista si fara sa fie pus intr-o combinatie) iar o combinatie trebuie sa aiba o sala . Legatura dintre cele doua tabele este realizata prin campul id_sala.

- **Constrangerile utilizate**

1. Primary key, folosita pentru a identifica in mod unic o inregistrare (campul nu poate fi nul):
    - **id_film** (entitatea FILM)
    - **id_combinatie** (entitatea COMBINATIE)
    - **id_sala** (entitatea SALA)
    - **id_bilet** (entitatea BILET).
    - **id_client** (entitatea CLIENT)
    - **id_tip**(entitatea TIP_BILET).
    - **id_actor**(entitatea ACTOR).

- **id_gen**(entitatea GEN_FILM).

2. Foreign key:
    - **Detalii_Film_Film_FK** (tabela DETALII_FILM) prin care se face legatura intre tabelele DETALII_FILM si FILM . Astfel se stocheaza detaliile pentru fiecare film.
    - **Combinatie_Film_FK** si **Combinatie_Sala_FK** (tabela COMBINATIE) prin care se face legatura intre tabelele FILM si COMBINATIE, respectiv SALA si COMBINATIE. Astfel se pun in COMBINATIE id-urile salilor si a filmelor ce vor fi programate.
    - **Bilet_Client_FK** (tabela BILET) prin care se face legatura intre tabelele CLIENT si BILET. Astfel se cunoaste id-ul clientului in campul pentru bilet.
    - **Bilet_Combinatie_FK** (tabela BILET) prin care se face legatura intre tabelele COMBINATIE si BILET. Astfel cunoastem id-ul combinatiei, a filmului si a salii necesare biletului.
    - **Bilet_Tip_Bilet_FK** (tabela BILET) prin care se face legatura intre tabelele TIP_BILET si BILET. Astfel cunoaste numele si reducerea care se aplica pretului, in functie de statutul clientului.
    - **Detalii_film_Actor_FK** (tabela DETALII_FILM) prin care se face legatura intre tabelele ACTOR si DETALII_BILET. Astfel se cunoaste actorul din detaliile filmului.
    - **Detalii_film_Gen_film_FK** (tabela DETALII_FILM) prin care se face legatura intre tabelele GEN_FILM si DETALII_FILM. Astfel se cunoaste genul filmului.

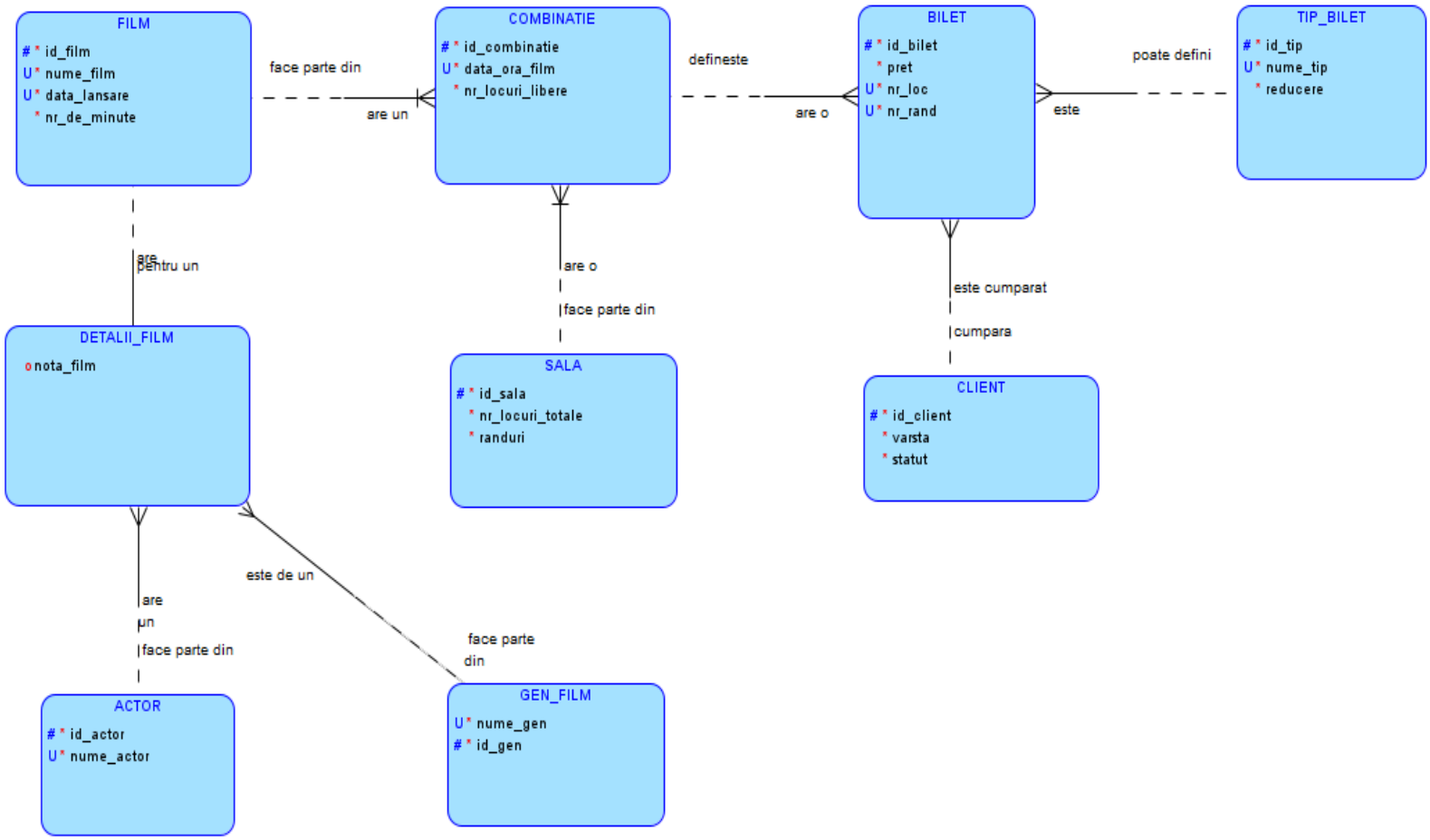
3. Unique key:
    - **Nume_Tip_UK** (din TIP_BILET): nu pot exista decat tipuri unice de bilet cu un pret fix.
    - **Bilet_Combinatie_Scaun_UK** (din BILET): nu trebuie sa exista 2 inregistrari cu acelasi loc, rand si combinatie .
    - **Combinatie_Sala_Ora_UK** (din COMBINATIE): verfica sa nu existe mai multe inregistrari cu aceeasi sala si aceeasi ora in acelasi timp.
    - **Film_Nume_Data_Lansare_UK** (din FILM): pot exista 2 filme cu acelasi nume dar sunt sanse foarte mici sa existe 2 filme cu acelasi nume si data in care a fost lansat.
    - **Nume_UK** (din ACTOR): nu pot exista actori cu mai multe id-uri.
    - **Nume_Gen_UK** (din GEN_FILM): nu pot exista decat genuri unice de film.

4. Check si trigger:

In mare parte sunt folosite verificari care asigura ca unele campuri sa apartina unui domeniu de valori (ca de exemplu pretul sa nu fie negativ, nr_loc sa fie mai mare decat 1 si mai mic decat 20, varsta sa fie strict mai mare decat 0, statutul sa faca parte dintr-o lista de tipuri la fel si tipul biletului).
Sunt prezente verificari de tip regex prin care se asigura validarea campurilor actor care nu trebuie sa contina numere .
Exista trigger pentru data_lansare pentru a ne asigura ca filmul nu va fi inregistrat pe o data mai veche de de 01.01.1922, in principiu, pentru a nu se face confuzie intre anul curent si cel al filmului.

5. Autoincrement
   Sistemele de autoincrement sunt folosite pentru cheile primare id din toate tabelele (id_film, id_combinatie, id_sala, id_bilet, id_client, id_tip, id_actor, id_gen) pentru a usura inregistrarea lor in sistem, privandu-ne de necesitatea de a le introduce manual. Acestea pornesc de la 1 si se incrementeaza cu 1.

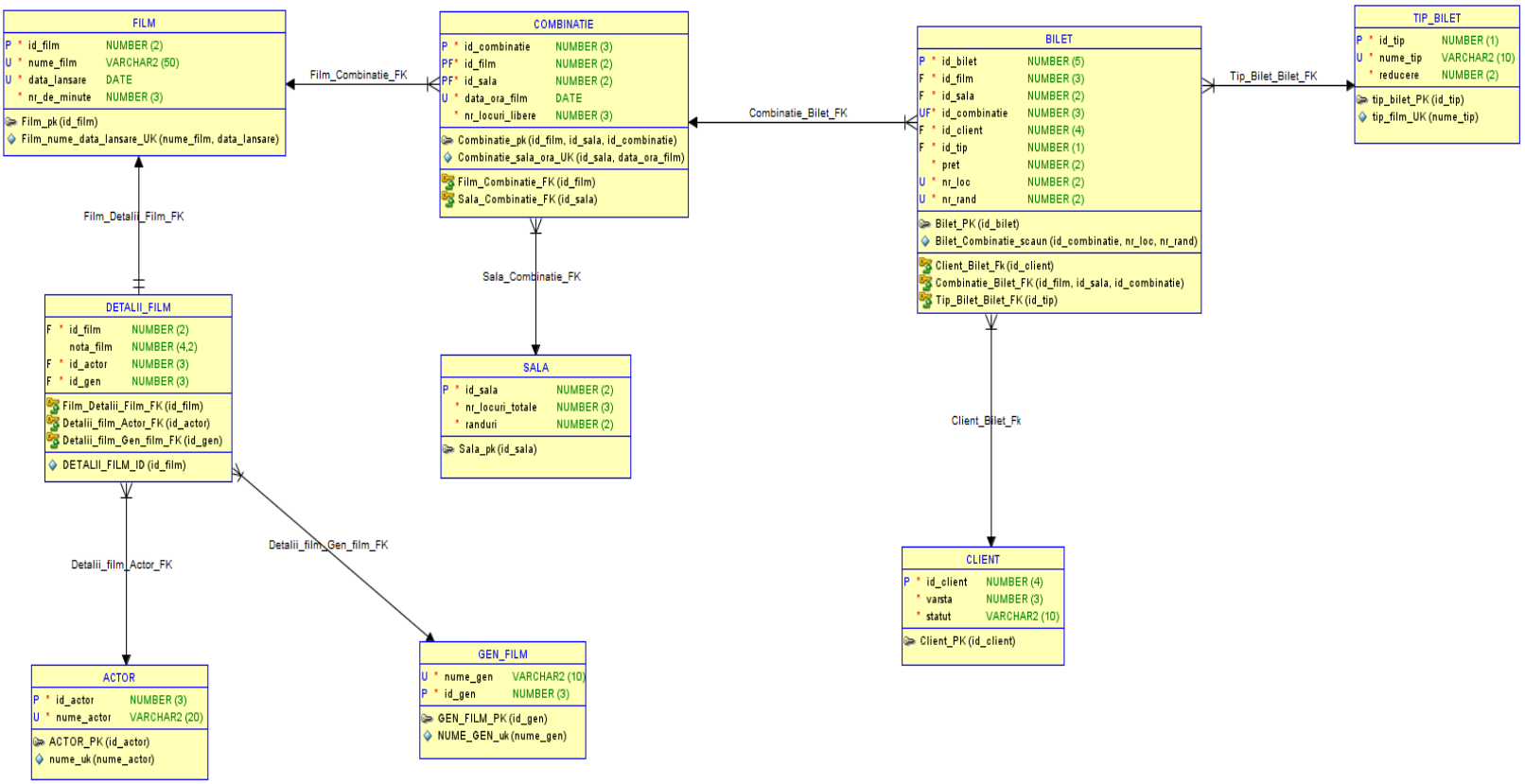
Modelul logic



In aceasta baze de date, toate entitatile indeplinesc primele trei niveluri de normalizare(mai putin tabela COMBINATIE care ajuta normalizarea intre entitatile SALA si FILM) deoarece :

- ✓ Un atribut contine valori atomice din domeniul sau (si nu grupuri de astfel de valori).
- ✓ Nu contine grupuri care se repeta.
- ✓ Toate atributele non-cheie depind in totalitate de toate cheile candidat.
- ✓ Toate atributele non-cheie sunt direct dependente de toate cheile candidat.

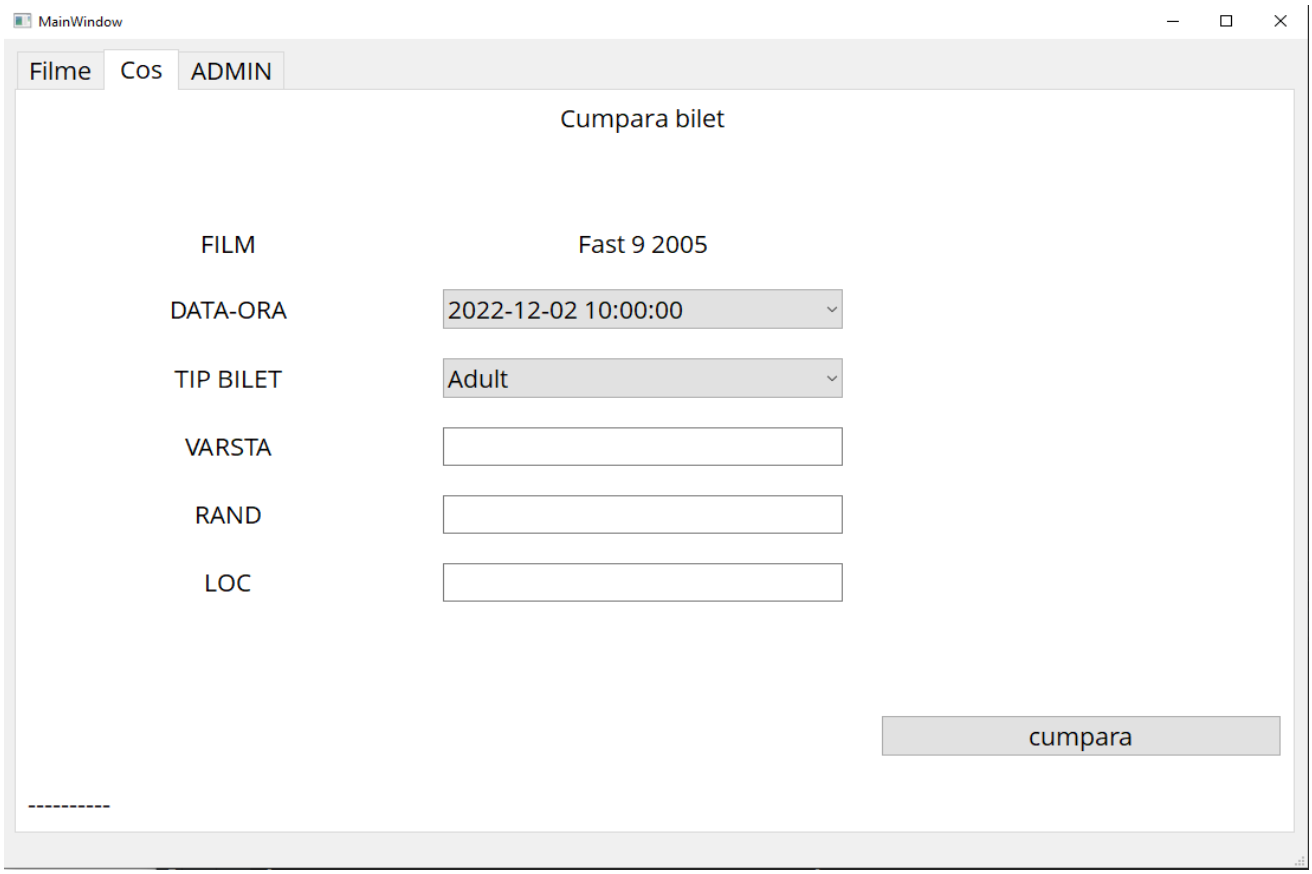
Modelul relational



- **Descrierea modalitatii de conectare la baza de date din aplicatie**

Pentru conectarea la baza de date am instalat pachetul cx_Oracle folosind comanda ’pip install cx_Oracle’ in terminal, am facut import librariei in codul aplicatiei si am folosit comanda connect() precizand host-ul, portul, baza de date, username-ul si parola necesare pentru a ma conecta la baza de date de pe serverul Oracle. La fiecare modificare a bazei de date s-a folosit comanda commit().

Aspect interfata pentru tabela BILET din punct de vedere al utilizatorului

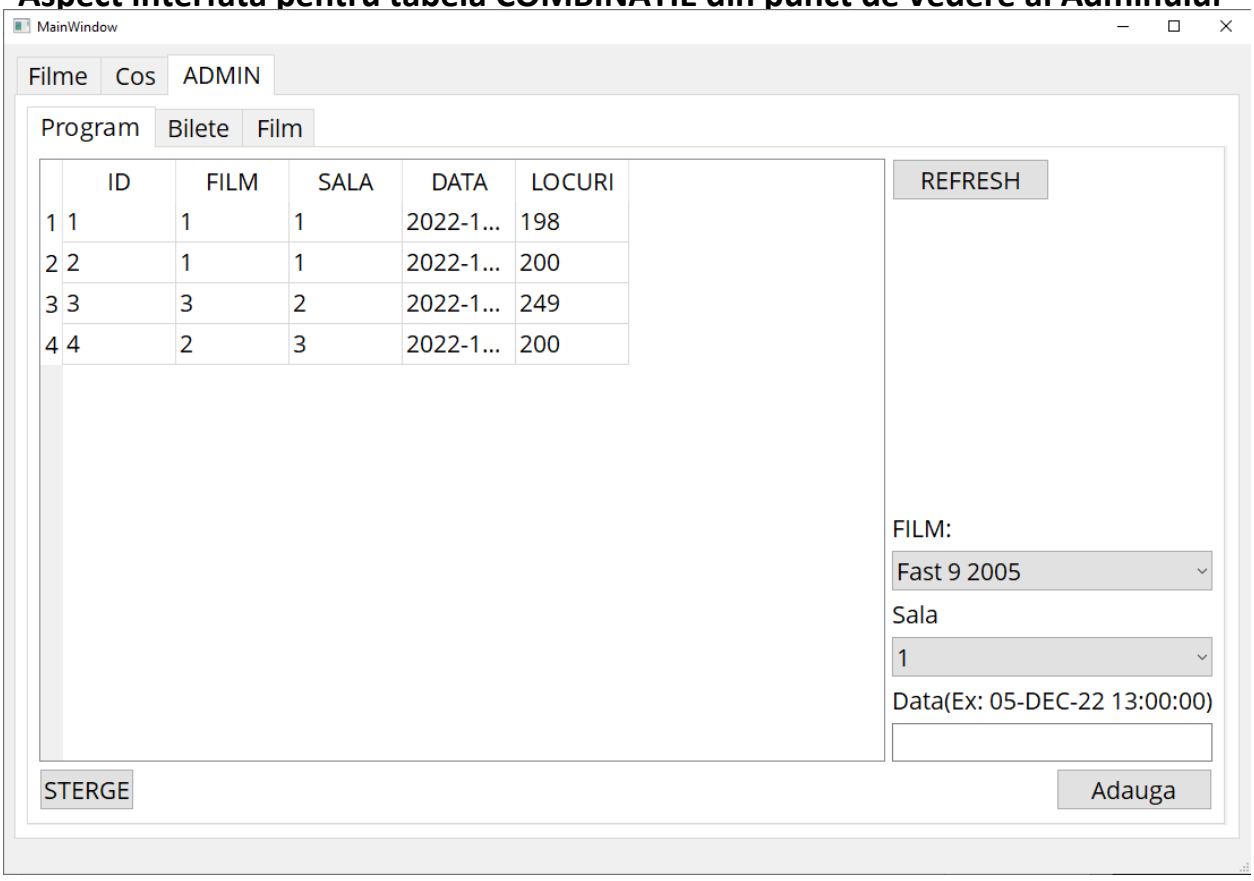


Metodele setupUI(self) si retranslateUI(self) din clasa UI_MainWindow creeaza interfata aplicatiei, respectiv redenumeste widget-urile folosite.

Metoda _buyBilet(self) din clasa UI_MainWindow se ocupa cu achizitia unui bilet, aceasta stocheaza inputul utilizatorului in diferite variabile cu ajutorul carora sunt integrate cheile primare si cheile foreign in tabela BILET:

```
def _buyBilet(self):
    if self.labelCosFilm.text()!="--":
        data_tmp = convert_date_format(self.comboBoxCosData.currentText()) ##data combinatiei, convertita gata de inserare
        nume_id_tip_tmp=self.comboBoxCosTip.currentText()                ##numele tipului de bilet=statut
        reducere_tmp=int(curs.execute("select reducere from tip_bilet where nume_tip=\'" + str(nume_id_tip_tmp) + "\'").fetchone()[0])
        id_tip_tmp=curs.execute("select id_tip from tip_bilet where nume_tip=\'" + str(nume_id_tip_tmp) + "\'").fetchone()[0] ##id-ul tipului biletului, de pus in tabela bilet
        varsta=int(self.lineEditVarsta.text())
        rand=int(self.lineEditCosRand.text())
        loc =int(self.lineEditCosLoc.text())

        ...
        id_comb = curs.execute("select id_combinatie from combinatie where id_film=" + str(self.id_film_buy) + " and data_ora_film=to_date(\'" + data_tmp + "\','DD-MON-YY HH24:MI:SS')").fetchone()[0]

        curs.execute("select * from bilet ")
        result = curs.fetchall()

        for row in result:  ##verific daca acelasi bilet e in tabel
            print(row[1],row[3],row[7],row[8])
            if row[1] == self.id_film_buy and row[3]==id_comb and row[7]==loc and row[8]==rand:
                QMessageBox.about(self.buttonAdaugaProgram, "Title", "Loc ocupat")
                return

        curs.execute("insert into client(varsta,statut) values("+str(varsta)+",\'"+nume_id_tip_tmp+"\')") ##inserez clientul
        conn.commit()

        pret = 25 - reducere_tmp
        id_client_tmp = curs.execute("SELECT id_client FROM client WHERE ROWID = (SELECT MAX(ROWID) FROM client)").fetchone()[0]

        #print("insert into bilet(id_film,id_sala,id_combinatie,id_client,id_tip,pret,nr_loc,nr_rand) values("+str(self.id_film_buy)+","+str(self.numar_sala_buy)+","+str(id_comb)+","+str(id_client_tmp)+
        curs.execute("insert into bilet(id_film,id_sala,id_combinatie,id_client,id_tip,pret,nr_loc,nr_rand) "
                     "values("+str(self.id_film_buy)+","+str(self.numar_sala_buy)+","+str(id_comb)+","+str(id_client_tmp)+","+str(id_tip_tmp)+","+str(pret)+","+str(loc)+","+str(rand)+")")
        conn.commit()
        curs.execute("update combinatie set nr_locuri_libere =nr_locuri_libere-1 where id_combinatie="+str(id_comb))
        conn.commit()
          conn.commit()

          pret = 25 - reducere_tmp
          id_client_tmp = curs.execute("SELECT id_client FROM client WHERE ROWID = (SELECT MAX(ROWID) FROM client)").fetchone()[0]

          #print("insert into bilet(id_film,id_sala,id_combinatie,id_client,id_tip,pret,nr_loc,nr_rand) values("+str(self.id_film_buy)+","+str(self.numar_sala_buy)+","+str(id_comb)+","+st
          curs.execute("insert into bilet(id_film,id_sala,id_combinatie,id_client,id_tip,pret,nr_loc,nr_rand) "
                       "values("+str(self.id_film_buy)+","+str(self.numar_sala_buy)+","+str(id_comb)+","+str(id_client_tmp)+","+str(id_tip_tmp)+","+str(pret)+","+str(loc)+","+str(rand)+")")
          conn.commit()
          curs.execute("update combinatie set nr_locuri_libere =nr_locuri_libere-1 where id_combinatie="+str(id_comb))
          conn.commit()

          QMessageBox.about(self.buttonAdaugaProgram, "Title", "Biletul achizitionat cu succes")
          id_bilet=curs.execute("SELECT id_bilet FROM bilet WHERE ROWID = (SELECT MAX(ROWID) FROM bilet)").fetchone()[0]
          self.labelAprobareAchzitie.setText("Biletul cu nr #"+str(id_bilet)+" |  "+str(nume_id_tip_tmp)+" | loc: "
              +""+str(loc)+" | rand: "+str(rand)+" | film: "+str(self.film_nume_buy)+" | data: "+str(data_tmp)+" a fost achizitionat cu succes")
    else:
        QMessageBox.about(self.buttonAdaugaProgram, "Title", "Alege un film boo")
```

Aspect interfata pentru tabela COMBINATIE din punct de vedere al Adminului



Asemanator cu functionalitatea de cumparare a unui bilet este si rolul butonului „Adauga” din tab-ul Program care stocheaza inputul adminului in diferite variabile cu ajutorul carora sunt integrate cheile primare si cheile foreign in tabela COMBINATIE:

```
def _adaugaProgram(self):
    data = 0
    if self.lineEditProgramData.text() and (self.comboBoxProgramFilm.currentText() != '-') and (
            self.comboBoxProgramSala.currentText() != '-'):
        data = self.lineEditProgramData.text()
        match = re.match(
            r'^(0[1-9]|[12][0-9]|3[01])-(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)-\d\d\s(2[0-3]|[01][0-9]):([0-5][0-9]):([0-5][0-9])$',
            data)
        if match:
            print("string ok ")
        else:
            QMessageBox.about(self.buttonAdaugaProgram, "Title1", "Data introdusa gresit")
            return
    else:
        QMessageBox.about(self.buttonAdaugaProgram, "Title", "Un camp e gol")
        return
    input_id_film = int(idFilme[self.comboBoxProgramFilm.currentIndex() - 1])
    input_id_sala = int(self.comboBoxProgramSala.currentText())
    nr_locuri = curs.execute("select nr_locuri_totale from sala where id_sala=" + str(input_id_sala)).fetchone()  # nr_locuri[0]

    curs.execute('select * from combinatie')
    result = curs.fetchall()
    for row in result:  ##verific daca aceeasi combinatie e in tabel
        if row[1] == input_id_film and row[2] == input_id_sala and data[7:9] == str(row[3])[2:4] and convertMonth(
                str(row[3])[5:7]) == data[3:6] and str(row[3])[8:10] == data[0:2] and str(row[3])[11:19] == data[
                                                                                                            10:18]:  # an luna zi timp
            QMessageBox.about(self.buttonAdaugaProgram, "Title", "Mai incearca")
            return
    print(input_id_film, input_id_sala, data)
    curs.execute(
        "insert into combinatie(id_film,id_sala,data_ora_film,nr_locuri_libere) values(:B ,:C ,to_date(:D,'DD-MON-YY HH24:MI:SS') ,:E)",
        (input_id_film, input_id_sala, data, nr_locuri[0]))
    conn.commit()
```