UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA BAZE DE DATE
-TEMĂ DE CASĂ-

Selecția unei echipe naționale de fotbal

Coordonator,

Ș.l.dr.ing. Cristian-Nicolae Buțincu

Student,
Melinte Alexandru-Gicu

Iași, 2021

Titlu proiect : Selecția unei echipe naționale de fotbal

Analiza, proiectarea și implementarea unei baze de date și a aplicației aferente care să modeleze selecția unei echipe naționale de fotbal, adică convocarea jucătorilor disponibili.

Descrierea cerintelor si modul de organizare al proiectului

Volumul mare de jucători disponibili pentru o națională determină necesitatea filtrării jucătorilor celor mai potriviți, eficienți și prolifici. Selecția unei echipe naționale implică o documentare vastă pentru selecționer întrucât are de luat în calcul diferite criterii de diferențiere. În primul rând, criteriul esențial de selecție este reprezentat de statutul de jucător activ sau retras, întrucât jucătorii retrași figurează în continuare în istoricul unei echipe de club. Un al doilea criteriu de filtrare este ca jucătorul să aibă un minim de meciuri jucate, impus de selecționer, iar pentru selecția fundașilor, mijlocașilor și atacanților este important și numărul de goluri sau o statistică ce scoate în evidență numărul de goluri per meci. Un al treilea și ultim criteriu de departajare este în cazul în care jucătorul disponibil dispune de un atuu, acesta având un avantaj față de cei care nu au.

Informațiile de care avem nevoie sunt cele legate de:

- ☺ **jucător_de_fotbal ->** ne interesează să știm cine este posibilul jucător selecționat (jucătorul are un ID, date personale, o poziție în teren și un atuu suplimentar, o statistică legată de numărul de meciuri jucate și numărul de goluri marcate și aparține de o echipă de club, evoluând într-o anumită ligă, este activ sau retras și are o naționalitate).
- ☺ **echipă_de_club ->** în cazul echipei de club ne interesează denumirea echipei, orașul reprezentat, lotul de jucători în format numeric și antrenorul principal. Echipele de club se diferențiază după un ID generat și numele echipei.
- ☺ **liga_de_fotbal ->** este alcătuită dintr-un număr de echipe de club, fiind disponibile doar primele ligi din diferite țări și un număr de competiții. Fiecare ligă este diferențiată după un ID.
- ☺ **competiție ->** fiecare competiție are un nume unic, un număr de trofee cu echipa(collective) obligatoriu și un număr de trofee individuale suplimentar și poate fi de două feluri: pentru echipele de club(specificând și liga) sau pentru echipele naționale.

☺ **trofee ->** acestea sunt câștigate de jucători în cadrul unei competiții cu echipa națională sau echipa de club ori individual în urma unei evoluții demne de un premiu. Acestea sunt adăugate la palmaresul fiecărui jucător. Fiecare trofeu are un ID unic și un an de triumf.

☺ **Palmares ->** ne interesează numărul de premii individuale și numărul de cupe câștigate. Fiecare jucător are propriul lui palmares unic.(după un ID).

☺ **Direcționare ->** un palmares poate conține mai multe trofee, iar un trofeu poate aparține mai multor jucători în cazul unui triumf colectiv. Așadar se impune o tabelă de direcționare între palmares și trofee reprezentată de ID-urile trofeului și palmaresului.

☺ **echipa_națională ->** o țară este reprezentată în mod unic de o echipă națională, astfel încât se impune ca o echipă națională să reprezinte doar o țară și nu mai multe. Echipa națională are un număr de jucători convocați, un selecționer și aparține de o anumită grupă națională.

☺ **Grupe_naționale ->** acestea sunt identificate unic după un ID ce denotă continentul din care face parte țara reprezentată de o echipă națională. Fiecare grupă are un continent disponibil cu un anumit număr de echipe naționale. Echipele de club nu pot face parte dintr-o grupă națională, așadar, pentru acest caz, avem un ID special.

Descrierea funcțională a aplicației

- Evidența jucătorilor de fotbal disponibili într-un club;
- Evidența echipelor de club disponibile dintr-o ligă de fotbal;
- Evidența competițiilor și trofeelor pentru liga de fotbal și pentru echipele naționale;
- Evidența echipelor naționalor în cadrul unor grupe naționale;
- Generarea palmaresului unui jucător prin intermediul trofeelor câștigate.

Descrierea detaliată a entităților și a relațiilor dintre tabele

Tabelele din aceasta aplicație sunt:

- jucător_de_fotbal;
- echipă_de_club;
- ligă_de_fotbal;
- competiție;
- trofee;
- Direcționare;
- Palmares;
- echipă_națională;
- grupe_naționale;

În proiectarea acestei baze de date s-au identificat tipurile de relații: one-to-one(1:1), one-to-many(1:n sau n:1) și many-to-many(m:n).

Între tabelele **jucător_de_fotbal** și **echipă_de_club** se observă o relație de tip **one-to-many(n:1)** deoarece un jucător de fotbal poate aparține doar unei echipe de club, iar o echipă de club conține mai mulți jucători. Legătura dintre cele două tabele este realizată prin câmpul **id_club**.

Între tabelele **jucător_de_fotbal** și **echipă_națională** se stabilește o relație de tip **one-to-many(n:1)** deoarece un jucător de fotbal poate reprezenta doar o echipă națională, iar o echipă națională convoacă mai mulți jucători. Legătura dintre cele două tabele este realizată prin câmpul **țară**.

Între tabelele **jucător_de_fotbal** și **Palmares** se regăsește o relație de tip **one-to-one(1:1)** deoarece un jucător de fotbal are un singur palmares, iar un palmares este atribuit doar unui jucător(doi jucători nu pot avea același palmares, cum nici același palmares nu poate fi impărțit de doi jucători).

Între tabelele **echipă_de_club** și **ligă_de_fotbal** se întâlnește o relație de tip **one-to-many(n:1)** deoarece o echipă de club poate aparține doar unei ligi de fotbal, iar o ligă de fotbal conține mai multe echipe de club. Legătura dintre cele două tabele este realizată prin câmpul **id_ligă**.

Între tabelele **ligă_de_fotbal** și **competiție** se observă o relație de tip **one-to-many(1:n)** deoarece o ligă de fotbal are mai multe competiții disponibile, pe când o competiție nu se poate desfășura în paralel în mai multe ligi(de exemplu: Liga 1 din România are Cupa României și campionatul, Liga 1 din Franța nu poate avea aceleași competiții). Legătura dintre cele două tabele este realizată prin câmpul **id_ligă**.

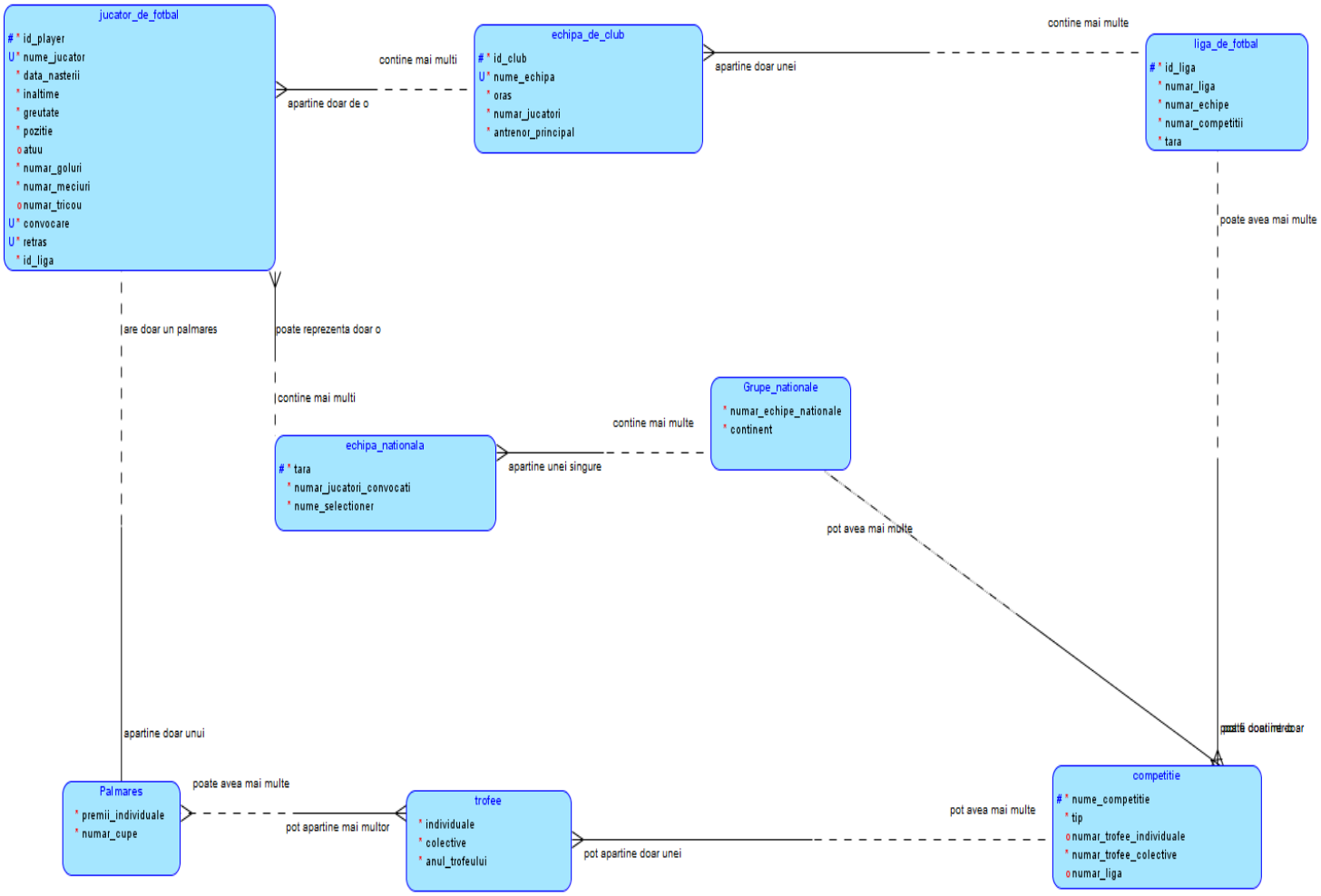
Între tabelele **competiție** și **Grupe_naționale** se stabilește o relație de tip **one-to-many(n:1)** deoarece o competiție nu poate fi, concomitent, în două grupe naționale, pe când o grupă națională poate avea mai multe competiții(de exemplu: Grupa Europei are Campionatul European si Liga Națiunilor, în Grupa Asiei nu poate exista Campionatul European). Legătura dintre cele două tabele este realizată prin câmpul **Grupe_naționale_ID**.

Între tabelele **Grupe_naționale** și **echipă_națională** se regăsește o relație de tip **one-to-many(1:n)** deoarece o grupă națională poate conține mai multe echipe naționale, dar o echipă națională nu poate face parte din mai multe grupe. Legătura dintre cele două tabele este realizată prin câmpul **Grupe_naționale_ID**.

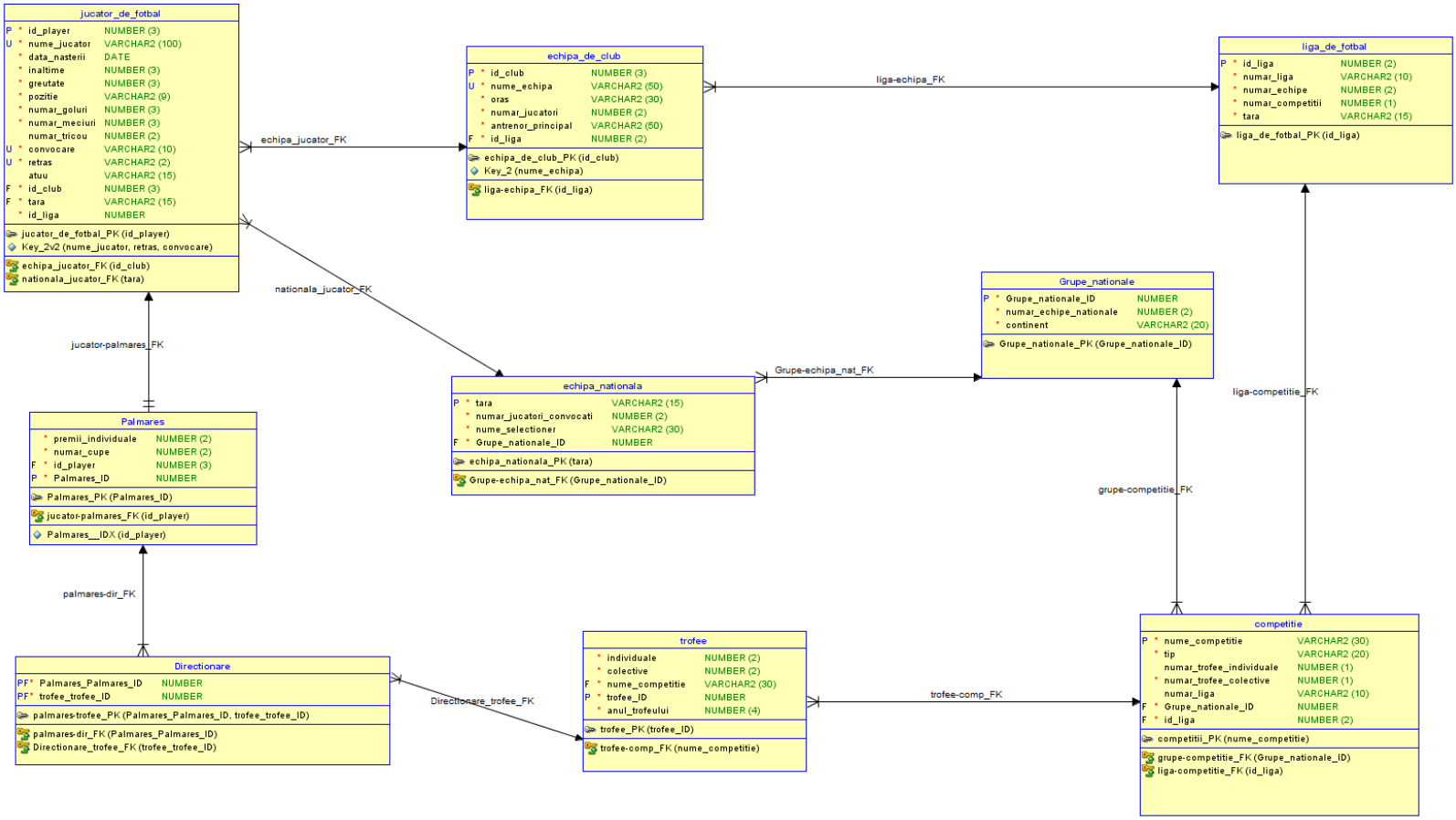
Între tabelele **competiție** și **trofee** se întâlnește o relație de tip **one-to-many(1:n)** deoarece o competiție poate conține mai multe trofee, dar același trofeu nu poate fi a mai

multor competiții. Legătura dintre cele două tabele este realizată prin câmpul **nume_competiție**.

Între tabelele **trofee** și **Palmares** se stabilește o relație **many-to-many(m:n)** deoarece un trofeu poate aparține mai multor palmarese, iar un palmares poate conține mai multe trofee(de exemplu: un trofeu colectiv se va atribui mai multor palmarese, iar unui palmares i se pot asocia mai multe trofee colective și individuale). Această relație se va sparge în două rezultând două relații **one-to-many(1:n)** și o nouă tabelă **Direcționare** care va include **trofee_ID** si **Palmares_ID**.



[Modelul logic al bazei de date]



Modelul relațional al bazei de date

Constrângerile utilizate

1) DE TIP CHECK:

- **nume_x(x poate fi jucător, echipă, competiție, antrenor, selecționer), țară, oraș** -> În afara numelui de echipă și competiție, care pot conține atât cifre cât și litere, restul numelor pot conține doar litere, fără alte caractere speciale; de asemenea acestea au și o lungime minimă;
- **data nașterii** -> va fi comparată cu SYSDATE pentru a nu fi o dată din viitor;
- **PK-urile(id_player, id_club, id_liga, Grupe_naționale_ID, Palmares_ID, trofee_ID)** -> nu pot fi mai mici decât 1 și sunt generate cu autoincrement;

- **poziție** -> poate lua valori doar din tupla {portar, fundaș, mijlocaș, atacant};
- **convocare** -> poate lua valori doar {convocat, neconvocat};
- **retras** -> poate lua valori {da, nu};
- **nume_ligă** -> valoare prestabilita “Liga 1”;
- **tip** -> poate lua valori doar {Echipă de club, Echipă națională};
- **înălțime, greutate, număr_x(x poate fi meciuri, goluri, trofee_individuale, trofee_colective, cupe), premii_individuale, individuale, colective** -> toate sunt impuse a fi pozitive, cu diferite valori de minim(0, 150, 50, 32 etc.);
- **număr_x(x poate fi jucători_convocați, jucători, echipe, competiții, tricou)** -> necesită a aparține unui anumit interval de valori.

2) DE TIP UNIQUE:

- Este necesar ca numele jucătorului sa fie unic, iar perechea (**nume_jucător, convocare, retras**) poate identifica orice jucător;
- Este necesar ca numele echipei să fie unic, întrucât nu pot exista două echipe cu același nume;
- De asemenea **primary-key-urile** sunt unice predefinite.

3) DE TIP NOT NULL:

- Cu excepția atributelor suplimentare(**număr_tricou, atuu, competiție.număr_trofee_indiviudale, competiție.număr_ligă**), restul atributelor sunt de tip NOT NULL.

Tehnologii utilizate pentru crearea proiectului:

- **Front-end:** pachetul tkinter (“Tk interface”) -> o legătură Python la setul de instrumente GUI Tk, reprezintă interfața standard Python pentru setul de instrumente Tk GUI și este de fapt GUI standard de Python.

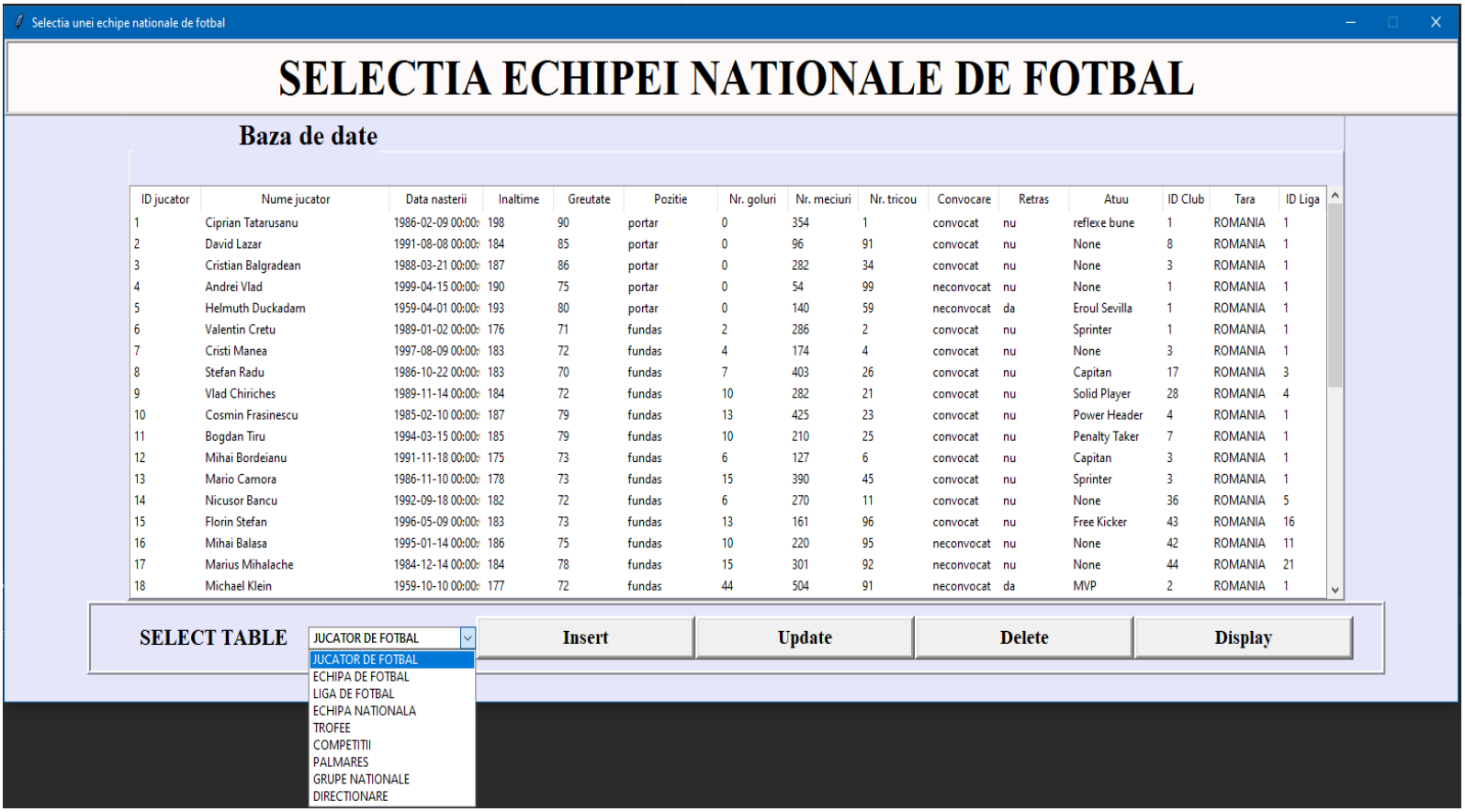
- **Back-end:** limbajul Python(3.9)
  - ☺ **SYS** -> modul ce oferă funcții și variabile care sunt utilizate la Runtime;
  - ☺ **RE(Regulat Expression Syntax)** -> modul ce oferă posibilitatea de a face match pe string-uri, la verificarea constrângerilor;
  - ☺ **Datetime** -> modul ce oferă disponibilitatea de a manipula datele de tip DATE;

- ☺ **Cx_Oracle** -> modul extensie care permite accesul Oracle Database și este compatibil cu Oracle Database 9.2, 10, 11, 12, 18 și 19. A fost testat pe mai multe versiuni de Pytthon, de la 3.6 la 3.9.
- ☺ **Tkinter** -> modul ce oferă diferite widget-uri pentru crearea unei interfețe(Label, Frame, LabelFrame, Combobox, Treeview, Entry, Scrollbar, Button)

Conectarea la baza de date:

Conectarea la baza de date se realizează prin intermediul modulului cx_Oracle utilizat în partea de back-end: connection = cx_Oracle.connect("alexm", "alexm", "localhost/xe").

Capturi de ecran concludente din interfață:



ID jucator	Nume jucator	Data nasterii	Inaltime	Greutate	Pozitie	Nr. goluri	Nr. meciuri	Nr. tricou	Convocare	Retras	Atuu	ID Club	Tara	ID Liga
1	Ciprian Tatarusanu	1986-02-09 00:00:	198	90	portar	0	354	1	convocat	nu	reflexe bune	1	ROMANIA	1
2	David Lazar	1991-08-08 00:00:	184	85	portar	0	96	91	convocat	nu	None	8	ROMANIA	1
3	Cristian Balgradean	1988-03-21 00:00:	187	86	portar	0	282	34	convocat	nu	None	3	ROMANIA	1
4	Andrei Vlad	1999-04-15 00:00:	190	75	portar	0	54	99	neconvocat	nu	None	1	ROMANIA	1
5	Helmuth Duckadam	1959-04-01 00:00:	193	80	portar	0	140	59	neconvocat	da	Eroul Sevilla	1	ROMANIA	1
6	Valentin Cretu	1989-01-02 00:00:	176	71	fundas	2	286	2	convocat	nu	Sprinter	1	ROMANIA	1
7	Cristi Manea	1997-08-09 00:00:	183	72	fundas	4	174	4	convocat	nu	None	3	ROMANIA	1
8	Stefan Radu	1986-10-22 00:00:	183	70	fundas	7	403	26	convocat	nu	Capitan	17	ROMANIA	3
9	Vlad Chiriches	1989-11-14 00:00:	184	72	fundas	10	282	21	convocat	nu	Solid Player	28	ROMANIA	4
10	Cosmin Frasinescu	1985-02-10 00:00:	187	79	fundas	13	425	23	convocat	nu	Power Header	4	ROMANIA	1
11	Bogdan Tiru	1994-03-15 00:00:	185	79	fundas	10	210	25	convocat	nu	Penalty Taker	7	ROMANIA	1
12	Mihai Bordeianu	1991-11-18 00:00:	175	73	fundas	6	127	6	convocat	nu	Capitan	3	ROMANIA	1
13	Mario Camora	1986-11-10 00:00:	178	73	fundas	15	390	45	convocat	nu	Sprinter	3	ROMANIA	1
14	Nicusor Bancu	1992-09-18 00:00:	182	72	fundas	6	270	11	convocat	nu	None	36	ROMANIA	5
15	Florin Stefan	1996-05-09 00:00:	183	73	fundas	13	161	96	convocat	nu	Free Kicker	43	ROMANIA	16
16	Mihai Balasa	1995-01-14 00:00:	186	75	fundas	10	220	95	neconvocat	nu	None	42	ROMANIA	11
17	Marius Mihalache	1984-12-14 00:00:	184	78	fundas	15	301	92	neconvocat	nu	None	44	ROMANIA	21
18	Michael Klein	1959-10-10 00:00:	177	72	fundas	44	504	91	neconvocat	da	MVP	2	ROMANIA	1

Image 1

În **“Image 1”** se poate vedea interfața ce afișează din baza de date tabela JUCATOR_DE_FOTBAL ce reține toate detaliile unui jucător de fotbal. Proiectul este axat pe naționala de fotbal a României dar se poate extinde pe orice națională se dorește. În interfață regăsim un combobox, unde se selectează tabela ce dorește a fi prelucrată, și patru butoane, fiecare cu o diferită funcționalitate; butonul de Insert face insert in baza de date, deschizând mai întâi încă o fereastră ce conține datele de completat și două butoane, unul de confirmare(OK) și unul de ștergere a datelor din câmpurile de completat(Clear); butonul de Update cu aceleași funcționalități ca cel de Insert cu diferența că acesta actualizează datele; butonul de Delete care șterge o linie din baza de date în momentul în care se face click pe PK-ul respectivei linii; butonul de Display care afișează baza de date.

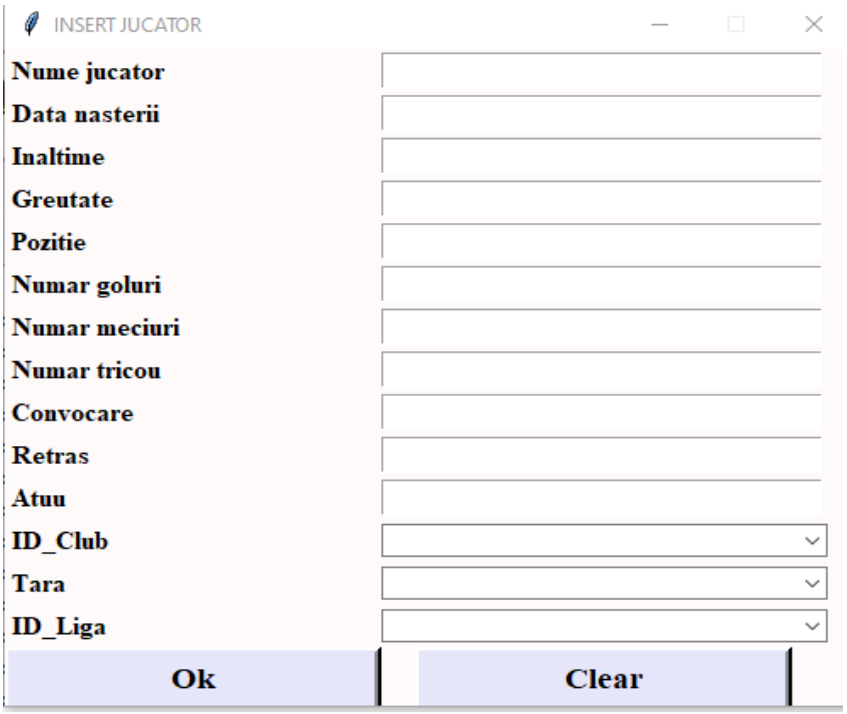


Image 2

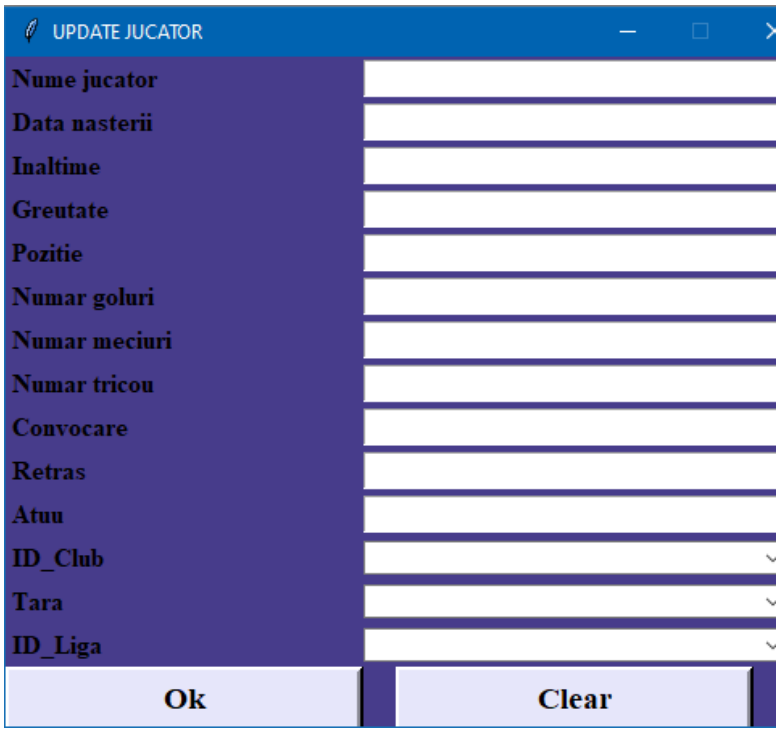


Image 3

În **“Image 2”** și **“Image 3”** se regăsește fereastra de insert în tabela JUCATOR_DE_FOTBAL și fereastra de update, iar în . **“Image 4”** codurile aferente funcțiilor dar și celei de delete.

```
    def addJucator(self):
        date1 = str(self.DataNasterii.get()).split('-')
        print(date1)
        print("Ziua")
        print(date1[0])
        print("Luna")
        print(int(date1[1]))
        print("Anul")
        print(date1[2])
        systemDate = date.today().strftime("%d-%m-%Y")
        print(systemDate)
        results = connection.cursor()
        command = "INSERT INTO jucator_de_fotbal VALUES(NULL, '%s', TO_DATE('%s', ' DD-MM-YYYY'), %s, %s, '%s', " \
                  "%s, %s, %s, '%s','%s','%s', %s," \
                  "'%s', %s)" % (
                      self.NumeJucator.get(), self.DataNasterii.get(), self.Inaltime.get(), self.Greutate.get(),
                      self.Pozitie.get(), self.NumarGoluri.get(), self.NumarMeciuri.get(), self.NumarTricou.get(),
                      self.Convocare.get(), self.Retras.get(), self.Atuu.get(),
                      self.txtID_Club_FK.get(), self.txtTara_FK.get(), self.txtID_Liga_FK.get())
        if self.verifyConstraints():
            print(command)
            results.execute(command)
            tkinter.messagebox.showinfo("Confirmare INSERT", "S-a adaugat!")
        else:
            tkinter.messagebox.showerror("Eroare INSERT", "Reintroduceti date valide!")
        connection.commit()

    def updateJucator(self):
        date1 = str(self.DataNasterii.get()).split('-')
        print(date1)
        print("Ziua")
        print(date1[0])
        print("Luna")
        print(date1[1])
        print("Anul")
        print(date1[2])
        systemDate = date.today().strftime("%d-%m-%Y")
        print(systemDate)
        results = connection.cursor()
        command = "UPDATE jucator_de_fotbal SET data_nasterii=TO_DATE('%s-%s-%s', ' DD-MM-YYYY'), inaltime=%s, " \
                  "greutate=%s, pozitie='%s', numar_goluri=%s, numar_meciuri=%s, numar_tricou=%s, atuu='%s', " \
```

```
def deleteJucator(self):
    results = connection.cursor()
    command = "DELETE FROM jucator_de_fotbal where id_player = %s" % self.getCellValue()
    results.execute(command)
    print(command)
    connection.commit()
    self.displayJucator()
    tkinter.messagebox.showinfo("Informatie",
                                "S-a efectuat stergerea liniei cu id_player = %s" % self.getCellValue())
```

Image 4

La nivel de interfață s-au implementat și constrângerile, la fiecare tabelă, asemenea celor din baza de date. Dacă datele sunt valide, la insert va apărea un mesaj de informare “S-a adaugat”, iar la update “S-a modificat înregistrarea“. În caz de nerespectare a constrângerilor, va apărea un mesaj de eroare cu textul “Reintroduceți date valide! “. La implementarea funcției de delete, în urma unei rulări corecte, va apărea un mesaj de confirmare “S-a efectuat ștergerea liniei cu PK = “. Un exemplu este în **“Image 5”,** iar implementarea este în **“Image 6”.**

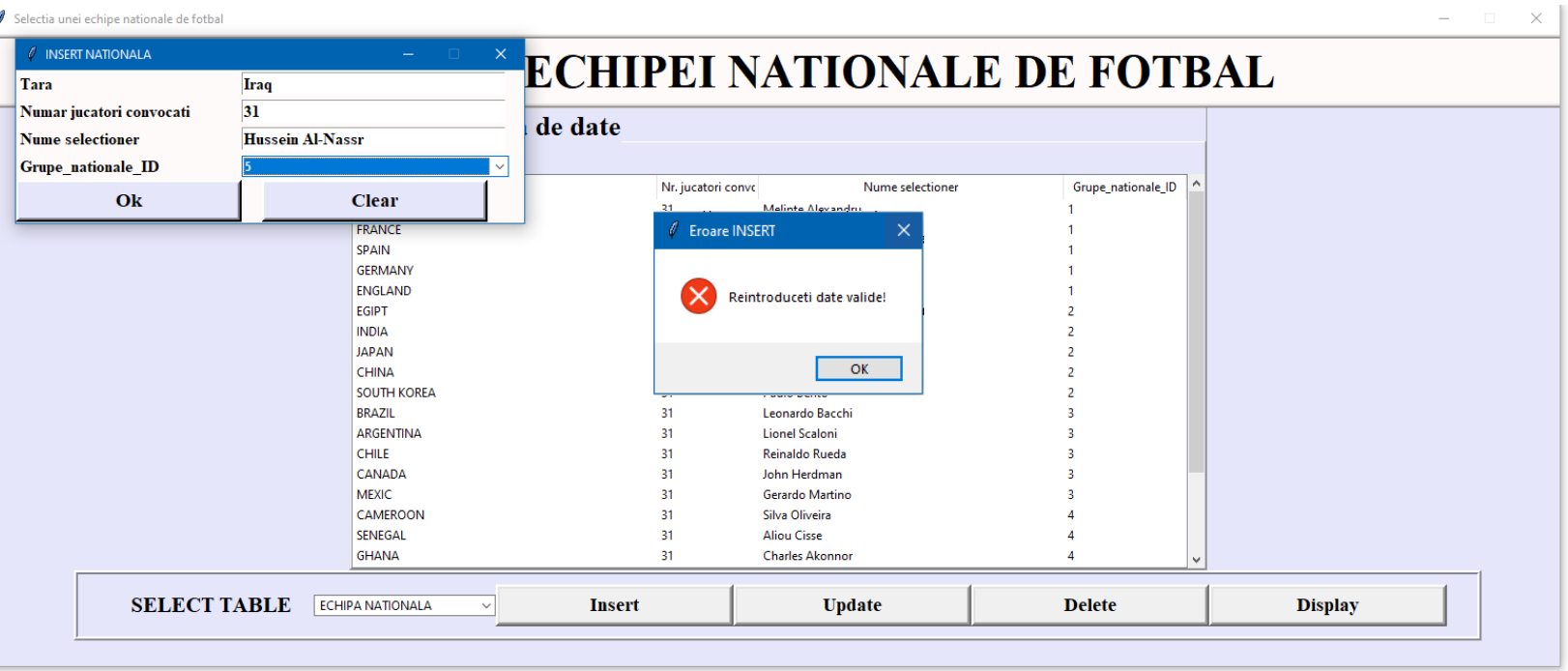


Image 5

```
def verifyConstraints(self):
    return (len(self.Tara.get()) >= 3 and re.match("^[A-Z]+[ A-Z]+$", self.Tara.get()) and
            int(self.NumarJucatoriConvocati.get()) >= 11 and int(
                self.NumarJucatoriConvocati.get()) <= 32 and re.match("[A-Z]{1}[A-Za-z-]+ [A-Z]{1}[A-Za-z-]+$",
                                                                      self.NumeSelectioner.get()) and self.txtGrupe_nationale_ID_FK.get() != "")

def addNationala(self):
    results = connection.cursor()
    command = "INSERT INTO echipa_nationala VALUES('%s', %s, '%s', %s)" % (
        self.Tara.get(), self.NumarJucatoriConvocati.get(), self.NumeSelectioner.get(),
        self.txtGrupe_nationale_ID_FK.get())
    if self.verifyConstraints():
        results.execute(command)
        tkinter.messagebox.showinfo("Confirmare INSERT", "S-a adaugat!")
    else:
        tkinter.messagebox.showerror("Eroare INSERT", "Reintroduceti date valide!")
    connection.commit()

def updateNationala(self):
    results = connection.cursor()
    command = "UPDATE echipa_nationala SET numar_jucatori_convocati = %s, nume_selectioner = '%s', Grupe_nationale_ID = %s WHERE tara = '%s'" % (
        self.NumarJucatoriConvocati.get(), self.NumeSelectioner.get(),
        self.txtGrupe_nationale_ID_FK.get(), self.Tara.get())
    if self.verifyConstraints():
        results.execute(command)
        tkinter.messagebox.showinfo("Confirmare", "S-a modificat inregistrarea!")
    else:
        tkinter.messagebox.showerror("Eroare UPDATE", "Reintroduceti date valide!")
    connection.commit()

def UpdateorInsert(self):
    if self.uoi == 1:
        return self.addNationala()
    if self.uoi == 2:
        return self.updateNationala()
```

Image 6