



**UNIVERSITATEA TEHNICĂ “GH ASACHI” IAȘI**  
**FACULTATEA AUTOMATICĂ ȘI CALCULATOARE**  
**SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI**  
**DISCIPLINA BAZE DE DATE PROIECT**

# **Cazarea studenților în căminele universitare pe perioada ciclului universitar**

**Coordonator,  
Prof. Cristian-Nicolae Buțincu**

**Student,  
Andrei Găină**

**Iași, 2020**

## Titlu proiect : Cazarea studentilor in caminele universitare pe perioada ciclului universitar

Analiza, proiectarea si implementarea unei baze de date si a aplicatiei aferente care sa modeleze procesul de cazare a studentilor in caminele universitatii pe parcursul ciclului universitar.

### Descrierea cerintelor si modul de organizare al proiectului

Volumul mare de studenti din cadrul unei universitati determină necesitatea unei aplicatii care va ajuta in procesul de cazare a acestora in caminele universitare, gestionarea acestui proces fiind o adevarata provocare. Se doreste a se implementa o aplicatie care va putea gestiona toti studentii care solicita cazare, repartizarea acestora in camine facandu-se dupa mai multi factori. In primul rand, cel mai important criteriu de cazare il reprezinta punctajul de cazare care ofera prioritate celor cu un punctaj mai mare fata de cei cu un punctaj mai scazut. Acest punctaj va fi calculat in mod obiectiv, in urma eforturilor depuse de fiecare student in cadrul procesului de acumulare si invatare a informatiilor. Un al doilea criteriu il va reprezenta optiunile studentilor. Fiecare student va putea opta pentru un anumit camin, o anumita camera si anumiți colegi insa repartizarea se va face in functie de punctaje. Un student cu un punctaj mai mare poate lua locul ales de un student cu un punctaj mai mic. Aplicatia va inregistra la nivelul bazei de date evidenta studentilor care solicita cazare, precum si date importante despre acestia(facultate din care fac parte, punctajele de cazare, actele depuse si optiunile acestora). In baza de date vom regasi caminele disponibile si camerele asociate acestora impreuna cu specificatiile esentiale. Se va mentine si evidenta administratorilor fiecarui camin in parte si tipurile de acte necesare intocmirii dosarului de cazare.

### Informatiile de care avem nevoie sunt cele legate de:

- **studenti:** ne intereseaza sa stim cine este studentul care solicita cazare (studentul este inscris la o anumita facultate, va avea asignat un cod matricol unic si de asemenea vom sti informatii exacte si despre domiciliul acestuia, numarul de telefon si optional email-ul)
- **facultati:** in cazul facultatilor ne intereseaza denumirea facultatii de la care provine studentul, adresa acesteia dar si un cod unic care identifica facultatea in cadrul universitatii;
- **administratorii:** sunt persoanele la conducerea caminelor care gestioneaza aceste cladiri; in cazul oricarei situatii aparute in interiorul caminelor pe parcursul cazarii, acestia vor fi persoanele care vor gestiona situatia; pentru a identifica un administrator vom avea nevoie de un nume, un numar de telefon si un email, urmand sa-i fie asignat un id unic;
- **caminele:** reprezinta multitudinea cladirilor in care vor fi cazati studentii; fiecare camin are o adresa si ofera informatii despre numarul de camere disponibile, acesta aflandu-se in grija unui administrator;
- **camere:** fiecare camera este identificata in mod unic prin codul caminului caruia apartine si numarul camerei din acel camin(perechea); attributele unei camere sunt reprezentate de numarul de locuri in camera, de tipul baii dar si de pretul de cazare;
- **optiuni:** fiecare student poate avea anumite optiuni in ceea ce priveste cazarea in camin; acestia pot solicita sa locuiasca intr-un anumit camin, intr-o anumita camera dar si cu anumiți colegi;
- **acte:** pentru a putea fi cazati, studentii trebuie sa depuna niste documente intr-un anumit interval al lunii specificate in care se fac cazarile; fiecare act este identificat printr-un numar unic acesta avand si o data la care a fost depus;
- **tipuri\_acte:** in cazul depunerii actelor exista un standard valabil in care se precizeaza tipurile de acte care trebuie aduse, unele dintre ele fiind optionale; de exemplu orice student trebuie sa depuna un contract de inchiriere si o copie de buletin dar nu orice student va putea depune adeverinta de membru intr-o asociatie;

- **punctaje:** fiecare student va avea un punctaj in functie de care vor fi prioritizate optiunile acestuia; astfel cei dintai carora li se va oferi cazarea in functie de optiunile alese vor fi cei care au un punctaj mai mare; acest punctaj este calculat in functie de media obtinuta in anul universitar precedent, in functie de numarul de credite acumulat pana in prezent dar si in functie de bonusul acordat in urma participarii la diferite concursuri sau a diferitelor activitati organizate de catre asociatiile studentesti in cadrul facultatii; pentru studentii de anul 1 se va lua in considerare doar media de la bacalaureat;

## Descrierea functionala a aplicatiei

Principalele funcții ale unei astfel de aplicatii sunt:

- ✓ Evidența studenților care solicită un loc la cămin
- ✓ Evidența opțiunilor acestora
- ✓ Evidența actelor depuse de aceștia
- ✓ Evidența căminelor si a caracteristicilor specifice (administratori, numar locuri in cămin, numar locuri in camere, tip baie)

## Tehnologii utilizate

### 1.Backend

Pentru partea de backend am ales sa utilizez limbajul **Python(3.7)** cu modulele urmatoare:

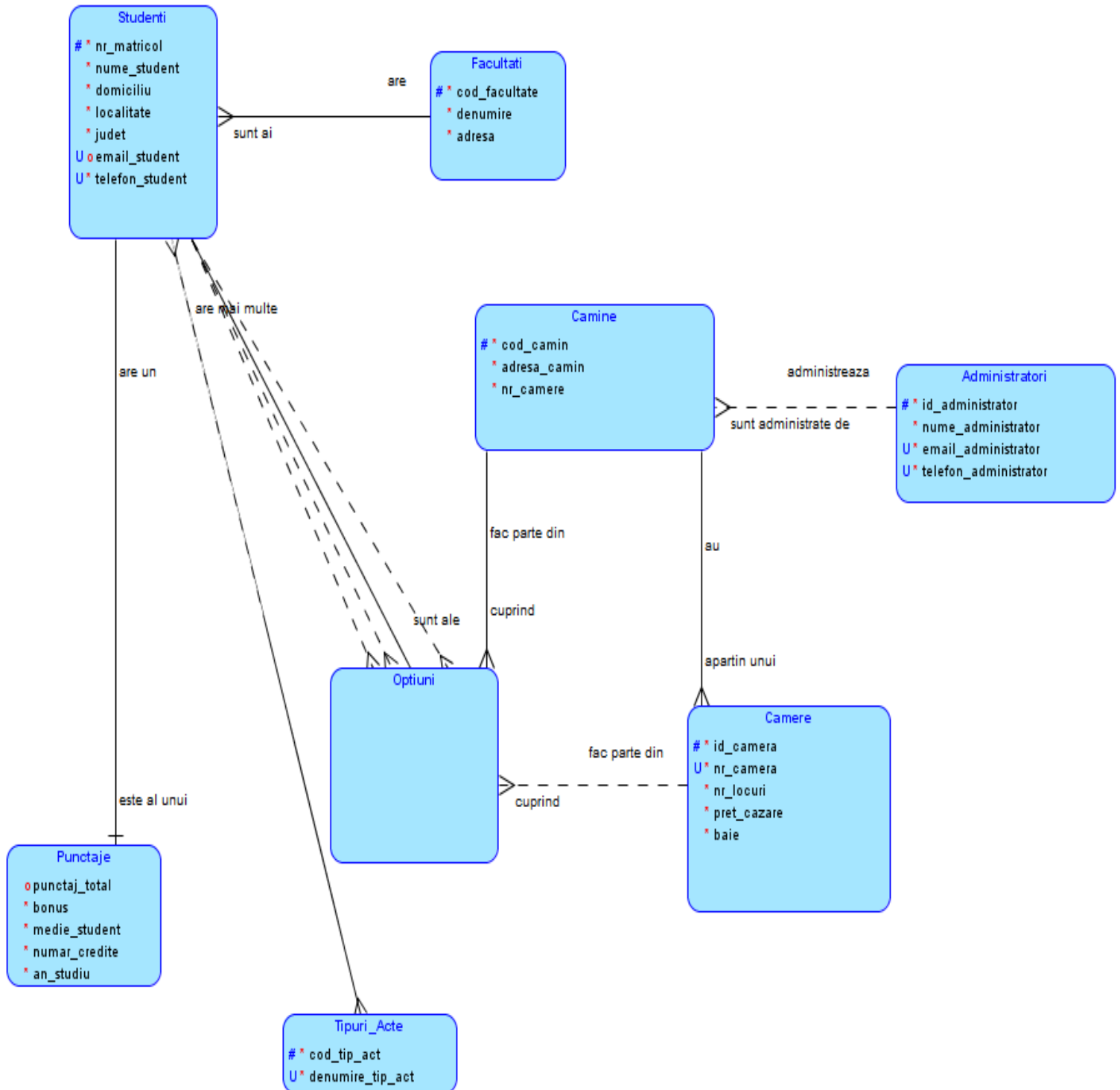
- **OS** – modul ce ajuta la interactiunea cu systemul de operare;
- **SYS** – modul ce ofera functii si variabile care sunt utilizare in manipularea a „**Python Runtime Environment**”;
- **Threading** – acest modul ajuta la rulara mai multor task-uri in paralel;
- **RE (Regular Expression Syntax)** – acest modul ne ajuta sa facem match pe anumite string-uri pentru a putea face diferite verificari a datelor inainte de a fi introduse in baza de date;
- **Datetime** – modul ce se detine functiile necesare manipularii datelor de tip DATE;
- **Cx\_Oracle** – este un modul extensie care permite accesul la **Oracle** Database si conform „Python database API specification” compatibil cu Oracle Database 9.2, 10, 11, 12, 18 si 19; ofera utilizarea completa a infrastructurii „Oracle Network Service”, inclusiv traficul de retea criptat si caracteristicile de securitate;
- **PyQt5** – este un set cuprinzator de legaturi Python pentru Qt v5; ajuta la crearea legaturilor dintre partea de backend si frontend;

### 2.Frontend

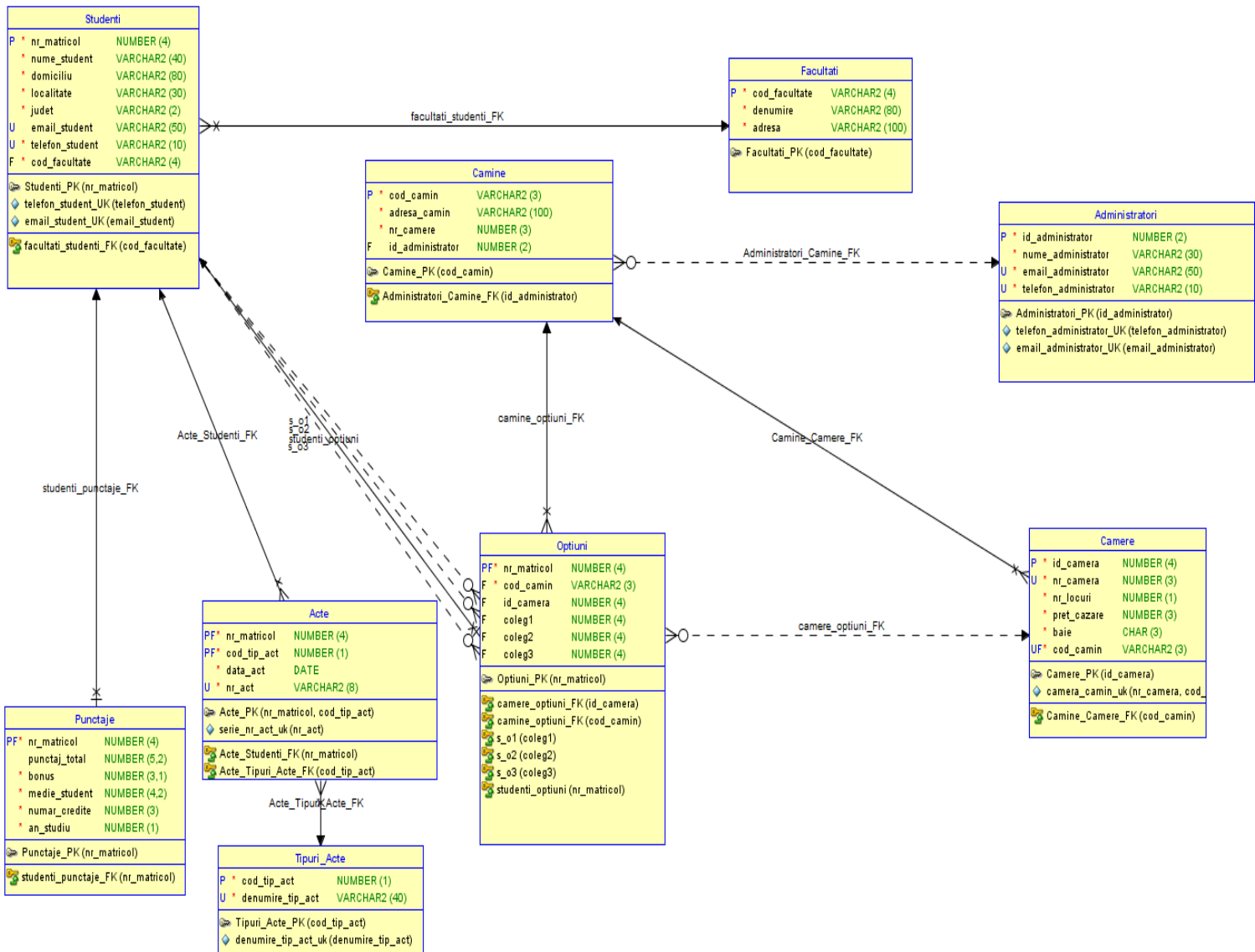
Pentru partea de frontend am ales sa utilizez **Qt Designer**.

Qt Designer este instrumentul Qt pentru proiectarea și construirea interfețelor grafice de utilizator (GUI) cu Qt Widgets.

## Schema logica a bazei de date



## Schema relationala a bazei de date



## Descrierea detaliată a entităților si a relațiilor dintre tabele

**Tabelele** din aceasta aplicație sunt:

- ✓ facultăți ;
- ✓ studenți ;
- ✓ administratori ;
- ✓ cămine ;
- ✓ camere ;
- ✓ opțiuni ;
- ✓ acte ;
- ✓ tipuri\_acte ;
- ✓ punctaje ;

In proiectarea acestei baze de date s-au identificat tipurile de **relatii 1:1 ,1:n, n:1 si n:m**.

Intre tabelele **studenti** si **punctaje** regasim o relatie de tip **1:1** deoarece un student are un punctaj(care se calculeaza cu ajutorul mediei, bonusului, numarului de credite si a anului de studiu in care se afla) iar un punctaj aparține unui singur student.

Intre tabelele **facultati** si **studenti** se întâlnește o relație de tip **1:n** deoarece o facultate poate avea mai multi studenti. Reciproca insa nu este valabila, deoarece un student poate solicita cazare la camin doar din partea unei facultati chiar daca acesta poate fi inscris la mai multe facultăți. Legătura dintre cele doua tabele este realizata prin campul **cod\_facultate**.

Intre **camine** si **camere** se stabileste o legatura de tip one-to-many. Tabela **camere** detine informatii despre camerele din camine in timp ce tabela **camine** contine toate caminele din campus asociate universitatii. Un camin are mai multe camere insa o camera apartine unui singur camin. Legatura dintre cele doua tabele este facuta de campul **cod\_camin**.

De asemenea intre tabela **camere** si tabela **optiuni** se stabileste o legatura de de tip **1:n**. Camerele se regasesc printre mai multe optiuni ale studentilor insa o optiune nu poate cuprinde mai multe camere. Legatura intre cele doua tabele se realizeaza prin campul **id\_camera**.

Intre **studenti** si **tipuri\_acte** se stabileste o legatura de tip many-to-many deoarece un student va depune mai multe tipuri de acte pentru a se putea caza si deasemenea acelasi tip de act se va regasi in randul a mai multor studenti(de exemplu Contractul de inchiriere il vor depune toti studentii). Aceasta relatie se va “sparge” in doua rezultand doua relatii **1:n** si o noua tabela **acte** care va tine evidenta fiecarui document depus de catre studenti, acest document avand un câmp specific **nr\_act** care-l identifica in mod unic.

Intre tabela **studenti** se stabilesc 4 legaturi de one\_to\_many cu tabela **optiuni**. Una dintre legaturi identifica optiunea completata de catre student in timp ce celelalte 3 legaturi identifica studentii alesi pentru a sta in camera cu respectivul solicitant. Legaturile intre cele 2 tabele se realizeaza cu ajutorul atributelor **nr\_matricol** (identifica solicitantul), **coleg1**, **coleg2**, **coleg3**( numere matricole care identifica studentii alesi ca si viitori colegi de camera).

De asemenea intre tabela **camine** si tabela **administratori** se stabileste o legatura de many\_to\_one. Un administrator poate administra mai multe camine in cadrul aceluiasi campus, insa un camin nu poate avea mai multi administratori la conducere(DOAR UNUL). Legatura intre cele doua se realizeaza prin campul **id\_administrator**.

Intre **camine** si **optiuni** se stabileste o legatura de tip one-to-many.O optiune poate cuprinde un singur camin in timp ce acelasi camin se poate afla printre optiunile a mai multor studenti. Legatura intre cele doua se realizeaza prin atributul **cod\_camin**.

### Constrangerile utilizate:

#### De tip check:

- MAIL- toate atributele care stocheaza adrese de mail au o constrangere pentru a ne asigura ca mail-ul introdus respecta formatul standard;
- TELEFON-toate atributele care stocheaza numere de telefon au o constrangere pentru a ne asigura ca introducem doar numere de telefon mobil valabile pe teritoriul Romaniei; nu se accepta numere de telefon fix;
- NUME – fiecare nume poate sa fi format doar din litere, cratima si spatiu fara cifre sau alte caractere speciale; de asemenea lungimea numelui trebuie sa fie mai mare de 2 litere;
- Primary key-urile nu pot fi mai mici decat 1 (in cazul numarului matricol mai mic decat 4000) si sunt generate cu autoincrement.
- JUDETELE- sunt reprezentate printr-un cod format din litere mari acest cod neputand contine mai putin de o litera sau mai mult de doua litere;
- COD\_FACULTATE:codul facultatilor este format din litere mari lungimea acestora fiind de minim un caracter si maxim 4 caractere;
- COD\_CAMIN- codul caminelor este format din litera „T” dar si din una sau doua cifre(T1-T21);
- Data actelor depuse va fi comparata cu SYSDATE pentru a nu fi o data din viitor si de asemenea aceasta trebuie sa se regaseasca intr-un anumit interval in care se pot depune actele pentru cazare( de exemplu 11-19 Septembrie).
- Numarul unic al actului este compus doar din cifre si litere.
- Baia dintr-un camin poate lua doar 3 valori: I-individuala , C2P-comuna pe palier si C2C-comuna la 2 camere;
- Media unui student nu poate fi mai mica decat 0(acesta are doar restante in cazul acesta) si mai mare decat 10.
- Numarul de credite acumulate este cuprins in intervalul 0-180.
- Bonusul nu poate fi mai mic decat 0 sau mai mare decat 10.
- Numarul de locuri in camera poate lua valori intre 1 si 4.

#### De tip Unique:

- Este necesar ca numarul de telefon si mailul studentilor dar si al administratorilor sa fie unice.
- De asemenea numele tipului de act dar si numarul actului vor fi unice.
- Prin perechea unica (nr\_camera, cod\_camin) vom putea identifica orice camera din cadrulcampusului.

#### De tip NOT NULL:

- NUME\_STUDENT este unul dintre atributele care are constrangerea NOT NULL deoarece avem nevoie sa cunoastem in cazul fiecarui student datele de identificare.
- De asemena telefonul administratorului dar si a studentului trebuie sa fie attribute de tipul NOT NULL deoarece avem nevoie sa îi identificam si sa îi contactam prin aceste numere de telefon.
- Marea majoritate a atributelor regasite in aceste tabele sunt de tip NOT NULL, deoarece acestea reprezinta informatii esentiale fara de care procesul de cazare nu poate avea loc.

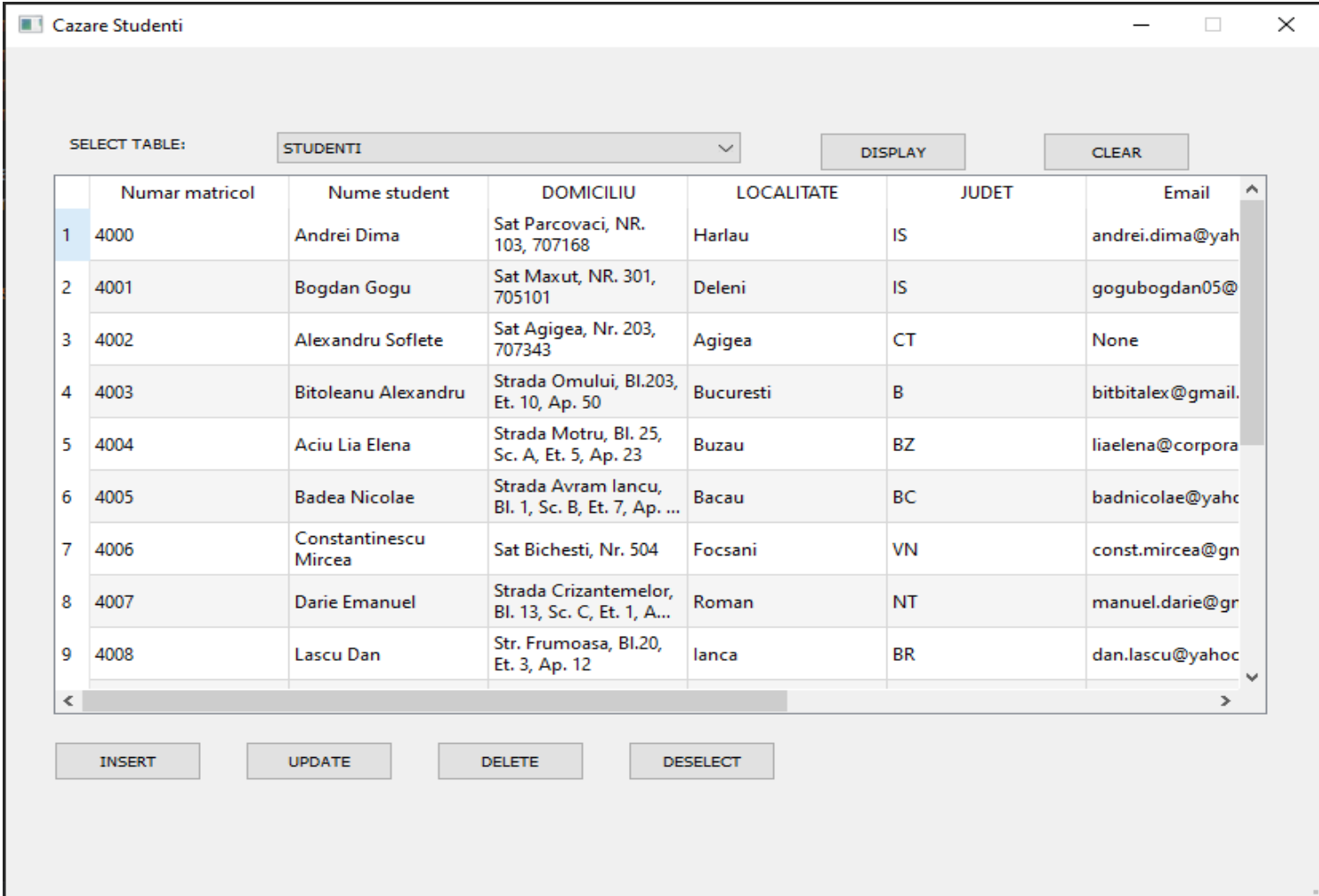
Exista si attribute care permit date de tip NULL. Un exemplu este email-ul unui student deoarece nu este obligatoriu ca toti studentii sa furnizeze un email. De asemenea punctajul total initial este NULL urmand sa fie calculat dupa introducerea factorilor de calcul. ID-ul administratorului din tabela camine poate lipsi pentru o scurta perioada, cazul acesta putand avea loc atunci cand un administrator îsi dă demisia. In tabela optiuni un student nu trebuie sa specifice in mod obligatoriu partenerii de camera.

## Conectarea la baza de date:

Conectarea la baza de date se realizeaza prin intermediul modulului cx\_Oracle utilizat pentru backend in tables.py: `connection = cx_Oracle.connect("andreibdp", "1234567", "localhost/xe")`

## Capturi de ecran:

In prima captura putem observa tabela **STUDENTI** care retine toate informatiile specifice si necesare cazarii unui student iar in cea de-a doua captura se poate vedea codul necesar implementarii acestei operatii de vizualizare.



The screenshot shows a web application window titled "Cazare Studenti". At the top, there is a "SELECT TABLE:" dropdown menu with "STUDENTI" selected, and two buttons: "DISPLAY" and "CLEAR". Below this is a table with 7 columns: "Numar matricol", "Nume student", "DOMICILIU", "LOCALITATE", "JUDET", and "Email". The table contains 9 rows of data. At the bottom of the window, there are four buttons: "INSERT", "UPDATE", "DELETE", and "DESELECT".

	Numar matricol	Nume student	DOMICILIU	LOCALITATE	JUDET	Email
1	4000	Andrei Dima	Sat Parcovaci, NR. 103, 707168	Harlau	IS	andrei.dima@yahoo.com
2	4001	Bogdan Gogu	Sat Maxut, NR. 301, 705101	Deleni	IS	gogubogdan05@yahoo.com
3	4002	Alexandru Soflete	Sat Agigea, Nr. 203, 707343	Agigea	CT	None
4	4003	Bitoleanu Alexandru	Strada Omului, Bl.203, Et. 10, Ap. 50	Bucuresti	B	bitbitalex@gmail.com
5	4004	Aciu Lia Elena	Strada Motru, Bl. 25, Sc. A, Et. 5, Ap. 23	Buzau	BZ	liaelena@corporatibuc.ro
6	4005	Badea Nicolae	Strada Avram Iancu, Bl. 1, Sc. B, Et. 7, Ap. ...	Bacau	BC	badnicolae@yahoo.com
7	4006	Constantinescu Mircea	Sat Bichesti, Nr. 504	Focsani	VN	const.mircea@gmail.com
8	4007	Darie Emanuel	Strada Crizantemelor, Bl. 13, Sc. C, Et. 1, A...	Roman	NT	manuel.darie@gmail.com
9	4008	Lascu Dan	Str. Frumoasa, Bl.20, Et. 3, Ap. 12	Ianca	BR	dan.lascu@yahoo.com

Captura 1



```

class Studenti:
    f = open("A:\Andrei\Facultate\AN3SEM1\BD\Tema\judete.txt", "r")

    def __init__(self):
        self.dialogBox = None
        self.errPopUp = None
        self.displayWindow = None
        self.judete = self.f.read().split("\n")
        self.f.close()

    @staticmethod
    def loadData(displayWindow):
        query = 'SELECT NR_MATRICOL \'Numar matricol \', NUME_STUDENT \'Nume student\', DOMICILIU, \' \
LOCALITATE, JUDET, EMAIL_STUDENT \'Email\', TELEFON_STUDENT \'Telefon\', \' \
COD_FACULTATE \'Cod facultate\' FROM studenti '
        col_cnt = "select count(*) from USER_TAB_COLUMNS where TABLE_NAME='STUDENTI'"
        results = connection.cursor()
        results.execute(col_cnt)
        for i in results:
            displayWindow.setColumnCount(i[0])
        results.execute(query)
        displayWindow.setRowCount(0)
        for row_number, row_data in enumerate(results):
            displayWindow.insertRow(row_number)
            for column_number, data in enumerate(row_data):
                displayWindow.setHorizontalHeaderItem(column_number,
                    QTableWidgetItem(str(results.description[column_number][0]))
                displayWindow.setItem(row_number, column_number, QTableWidgetItem(str(data)))

```

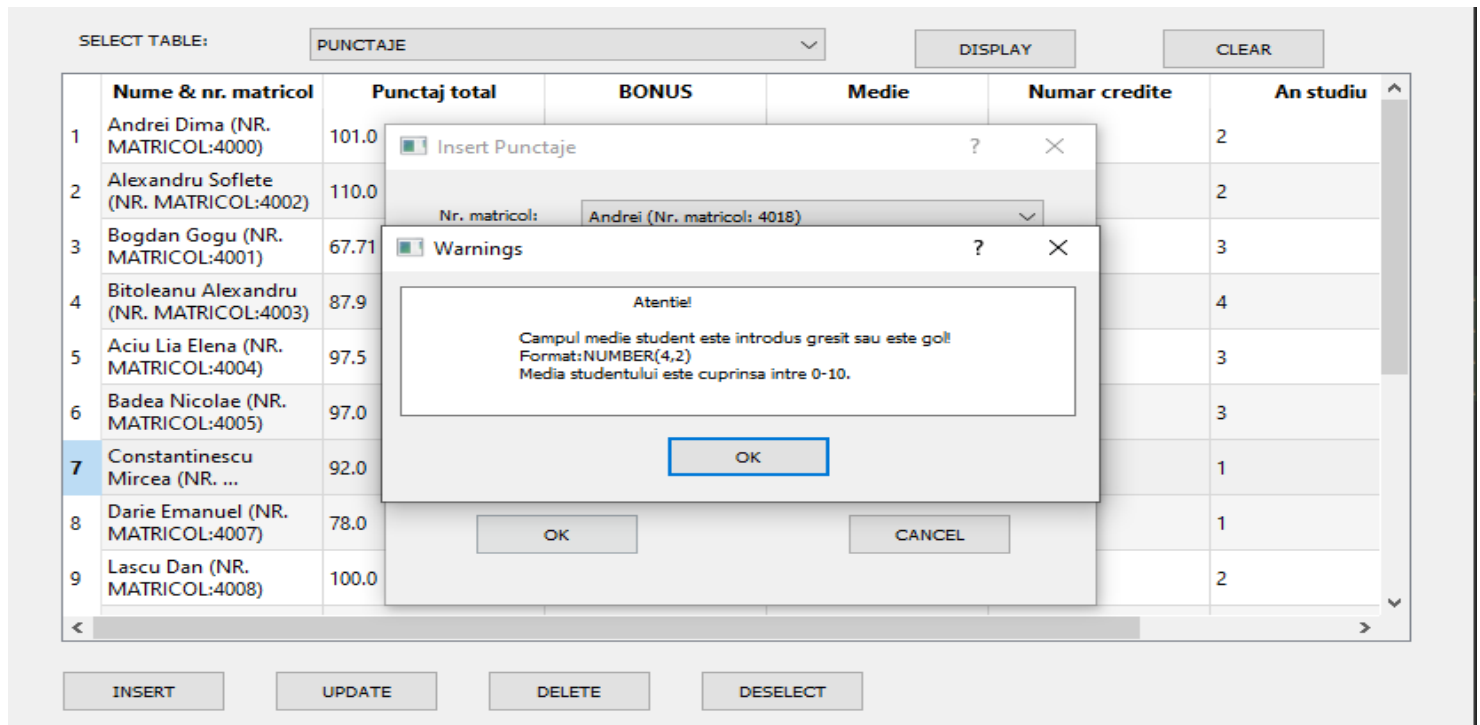
## Captura 2

In prima captura de mai jos putem vizualiza fereastra de **INSERT** necesara introducerii unei noi inregistrari in tabela **STUDENTI** iar in continuare putem analiza implementarea in cod a acestei operatii dar si a optiei de **DELETE**.

The screenshot shows a window titled "Cazare Studenti". At the top, there is a "SELECT TABLE:" dropdown menu with "STUDENTI" selected, and "DISPLAY" and "CLEAR" buttons. Below this is a table with columns: "Numar matricol", "Nume student", "DOMICILIU", "LOCALITATE", "JUDET", and "Email". The table contains 9 rows of data. Overlaid on the table is a dialog box titled "Insert Studenti". The dialog box has input fields for "NUME:", "FACULTATE:" (with a dropdown menu showing "AC"), "DOMICILIU:", "LOCALITATE:", "TELEFON:", "EMAIL:", and "JUDET:" (with a dropdown menu showing "B"). There are "OK" and "CANCEL" buttons at the bottom of the dialog box. At the bottom of the main window, there are buttons for "INSERT", "UPDATE", "DELETE", and "DESELECT".

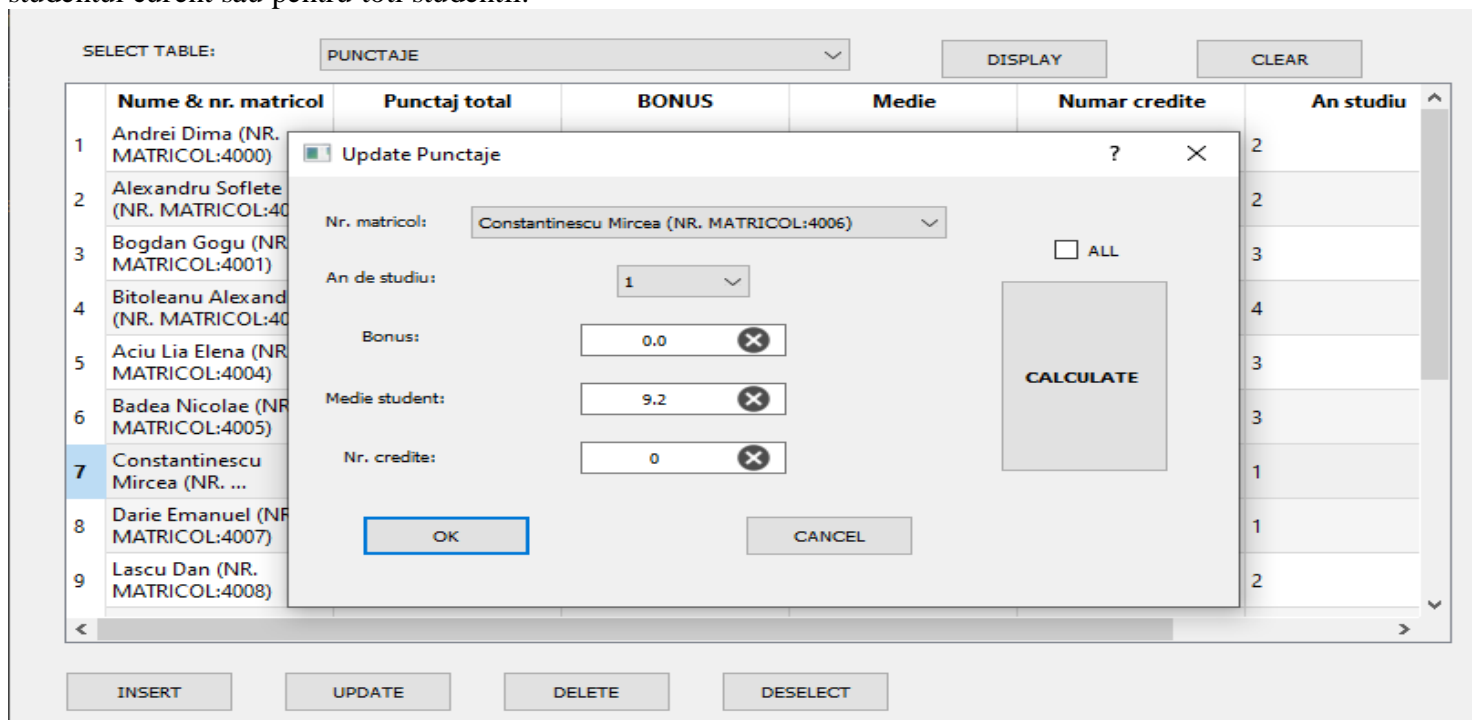
```
def verifyData(self):  
    results = connection.cursor()  
    judet = self.dialogBox.judetBox.currentText()  
    cod_facultate = self.dialogBox.facultateBox.currentText()  
    nume = re.search("^[A-Z][A-Za-z ,.-]+$", self.dialogBox.numeEdit.text())  
    domiciliu = re.search("^[-A-Za-z., 0-9-]+$", self.dialogBox.domiciliuEdit.toPlainText())  
    localitate = re.search("^[A-Z][-A-Za-z-]+$", self.dialogBox.localitateEdit.text())  
    telefon = re.search("(07)\d{8}$", self.dialogBox.telefonEdit.text())  
    email = re.search("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]+\.[a-z]{2,4}", self.dialogBox.emailEdit.text())  
    telefon_uk_ck = 1  
    if telefon is not None:  
        telefon_uk = "SELECT TELEFON_STUDENT FROM STUDENTI WHERE TELEFON_STUDENT = '%s'" % telefon.string  
        telefon_uk = results.execute(telefon_uk)  
        for i in telefon_uk:  
            if i:  
                telefon_uk_ck = 0  
  
    email_uk_ck = 1  
    if email is not None:  
        email_uk = "SELECT EMAIL_STUDENT FROM STUDENTI WHERE EMAIL_STUDENT = '%s'" % email.string  
        email_uk = results.execute(email_uk)  
        for i in email_uk:  
            if i:  
                email_uk_ck = 0  
  
    if nume is not None and \\\n        domiciliu is not None and \\\n        localitate is not None and \\\n        telefon is not None and \\\n        telefon_uk_ck and email_uk_ck:  
        if email is not None:  
            command = "INSERT INTO STUDENTI VALUES (%s,%s,%s,%s,%s,%s,%s,%s)" % ('NULL',\\\n                                                                    nume.string,\\\n                                                                    domiciliu.string,\\\n                                                                    localitate.string,\\\n                                                                    str(judet),\\\n                                                                    email.string,\\\n                                                                    telefon.string,\\\n                                                                    str(cod_facultate))  
  
            results.execute(command)  
            results.execute('COMMIT WORK') # tranzactie?  
            self.dialogBox.close()  
        else:  
            if telefon_uk_ck == 0:  
                self.errPopUp.show()  
                self.errPopUp.eroareEdit.setPlainText("\\t\\tAtentie!\\n\\n                    "  
                                                        "\\nMai exista un student cu acelasi numar de telefon!")  
            if email_uk_ck == 0:  
                self.errPopUp.eroareEdit.setPlainText("  
                "\\t\\tAtentie!\\n\\n                    Mai exista un student cu acelasi numar de telefon!"  
                "\\nDe asemenea exista un student si cu aceeași adresa de email!")  
            elif email_uk_ck == 0:  
                self.errPopUp.show()  
                self.errPopUp.eroareEdit.setPlainText("\\t\\tAtentie!\\n\\n                    "  
                                                        "\\nMai exista un student cu aceeași adresa de email!")  
            else:  
                self.errPopUp.show()  
                self.errPopUp.eroareEdit.setPlainText("\\t\\tAtentie!\\n\\n\\tAti uitat sa completati un camp cu date!"  
                                                        "\\n         Este posibil de asemenea sa fi completat un camp gresit!")  
  
@staticmethod  
def delete(displayWindow, errPopUp):  
    results = connection.cursor()  
    currentRow = displayWindow.currentRow()  
    data = []  
    for i in displayWindow.selectedItems():  
        data.append(i.text())  
    if currentRow != -1:  
        command = "DELETE FROM STUDENTI WHERE NR_MATRICOL= %s" % data[0]  
        results.execute(command)  
        displayWindow.removeRow(currentRow)  
        results.execute('COMMIT WORK') # tranzactie?  
    else:  
        errPopUp.show()  
        errPopUp.eroareEdit.setPlainText("\\t\\tAtentie!\\n\\n                    "  
                                          "\\nTrebuie sa selectati un row din tabela!")
```

In captura 5 putem observa si o fereastra de warning-uri.



## CAPTURA 5

In continuare vom putea vedea fereastra de update necesara pentru tabela **PUNCTAJE** si de asemenea putem urmări codul ce implementează atât **UPDATE**-ul pentru informațiile necesare calculării punctajelor cât și codul ce implementează calculul efectiv al acestor punctaje având posibilitatea să alegem dacă dorim să calculăm doar pentru studentul curent sau pentru toți studenții.



## CAPTURA 6

```

def verifyAndUpdate(self):
    results = connection.cursor()
    data = None
    query = 'SELECT * FROM PUNCTAJE WHERE NR_MATRICOL=%s' \
            % int(re.sub("[a-zA-Z.(): ]+", '', self.dialogBox.nrMatricolBox.currentText()))
    results.execute(query)
    self.errPopUp.eroareEdit.setPlainText("")
    for i in results:
        data = i
    self.modifications = False
    command = "UPDATE PUNCTAJE SET "
    if self.dialogBox.anStudiuBox.currentText() != str(data[5]):
        self.modifications = True
        an_studiu = self.dialogBox.anStudiuBox.currentText()
        command += "AN_STUDIU = %s," % int(an_studiu)
    if self.dialogBox.anStudiuBox.currentText() != 1:
        if self.dialogBox.bonusEdit.text() != str(data[2]):
            bonus = re.search('^([0-9]+\.[0-9])$', self.dialogBox.bonusEdit.text())
            if (bonus is None and int(self.dialogBox.anStudiuBox.currentText()) != 1) \
                or (bonus is not None and float(bonus.string) > 10):
                self.errPopUp.show()
                self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
                self.errPopUp.eroareEdit.insertPlainText("\t\tAtentie!\n\n"
                    "\tCampul bonus este introdus gresit sau este gol!\n"
                    "\tFormat:NUMBER(3,1)\n"
                    "\tBonusul este cuprins intre 0-10.\n"
                    "\tCampul bonus nu va fi actualizat.")
            elif bonus is not None:
                self.modifications = True
                command += "BONUS = %s," % float(bonus.string)
        if self.dialogBox.nrCrediteEdit.text() != str(data[4]):
            nr_credite = re.search('^([0-9]+$)', self.dialogBox.nrCrediteEdit.text())
            if (nr_credite is None and int(self.dialogBox.anStudiuBox.currentText()) != 1) or (
                nr_credite is not None
                and int(nr_credite.string) > (
                    60 * (int(self.dialogBox.anStudiuBox.currentText()) - 1))):
                self.errPopUp.show()
                self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
                self.errPopUp.eroareEdit.insertPlainText("\t\tAtentie!\n\n"
                    "\tCampul numar credite este introdus gresit sau este "
                    "gol!\n"
                    "\tFormat:NUMBER(3)\n"
                    "\tNumarul de credite este cuprins intre 0-180, in functie "
                    "de anul de studiu.\n"
                    "\tCampul nr_credite nu va fi actualizat.")
            elif nr_credite is not None:
                self.modifications = True
                command += "NUMAR_CREDITE = %s," % float(nr_credite.string)
        if self.dialogBox.medieStudentEdit.text() != str(data[3]):
            medie_student = re.search('^([0-9]+\.[0-9]+$)', self.dialogBox.medieStudentEdit.text())
            if medie_student is None or (medie_student is not None and float(medie_student.string) > 10):
                self.errPopUp.show()
                self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
                self.errPopUp.eroareEdit.insertPlainText("\t\tAtentie!\n\n"
                    "\tCampul medie student este introdus gresit sau este gol!\n"
                    "\tFormat:NUMBER(4,2)\n"
                    "\tMedia studentului este cuprinsa intre 0-10.")
            elif medie_student is not None:
                self.modifications = True
                command += "MEDIE_STUDENT = %s," % float(medie_student.string)
    if not self.modifications:
        self.errPopUp.show()
        self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
        self.errPopUp.eroareEdit.insertPlainText("\n\t\tAtentie!\n\n"
            "Nu s-a modificat niciun camp referitor"
            "la informatiile necesare calcularii punctajelor.")
    # self.dialogBox.close()
else:
    command = command[:-1] + " WHERE NR_MATRICOL = %s" \
            % int(re.sub("[a-zA-Z.(): ]+", '', self.dialogBox.nrMatricolBox.currentText()))
    results.execute(command)
    results.execute('COMMIT WORK') # tranzactie?
    # self.dialogBox.close()
    self.errPopUp.show()
    self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
    self.errPopUp.eroareEdit.insertPlainText("\n\t\tAtentie!\n\n"
        "Modificarea informatiilor s-a efectuat cu succes!")

```

```

def calculatePunctaj(self):
    results = connection.cursor()
    if self.dialogBox.checkBox.isChecked():
        command = 'UPDATE punctaje ' \
            'SET punctaj_total = CASE ' \
            'WHEN an_studiu=1 THEN (medie_student * 10) ' \
            'ELSE ' \
            '(medie_student * (numar_credite*10/(60.0*(an_studiu-1)))+ (bonus)) ' \
            'END'

        self.verifyAndUpdate()
        results.execute(command)
        results.execute('COMMIT WORK') # tranzactie?
        self.loadData(self.displayWindow)
        self.errPopUp.show()
        self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
        self.errPopUp.eroareEdit.insertPlainText("\n\t\tAtentie!\n\n"
            "Modificarea punctajelor s-a realizat cu succes pentru toti "
            "studentii!")
    else:
        command = 'UPDATE punctaje ' \
            'SET punctaj_total = CASE ' \
            'WHEN an_studiu=1 THEN (medie_student * 10) ' \
            'ELSE ' \
            '(medie_student * (numar_credite*10/(60.0*(an_studiu-1)))+ (bonus)) ' \
            'END where NR_MATRICOL=%s' \
            % int(re.sub("[a-zA-Z.()]: ", '', self.dialogBox.nrMatricolBox.currentText()))

        self.verifyAndUpdate()
        results.execute(command)
        results.execute('COMMIT WORK') # tranzactie
        self.loadData(self.displayWindow)
        self.errPopUp.show()
        self.errPopUp.eroareEdit.moveCursor(QTextCursor.End)
        self.errPopUp.eroareEdit.insertPlainText("\n\t\tAtentie!\n\n"
            "Update-ul punctajului curent s-a realizat cu succes!")

```

Captura 8