UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA:Tehnologia informației

Sistem de gestiune al studentilor unei facultati

Proiect la disciplina
Baze de Date

Cadru didactic coordonator:
Mironeanu Cătălin

Studenți:
Ruxandari Razvan 1307B

Iași, 2024

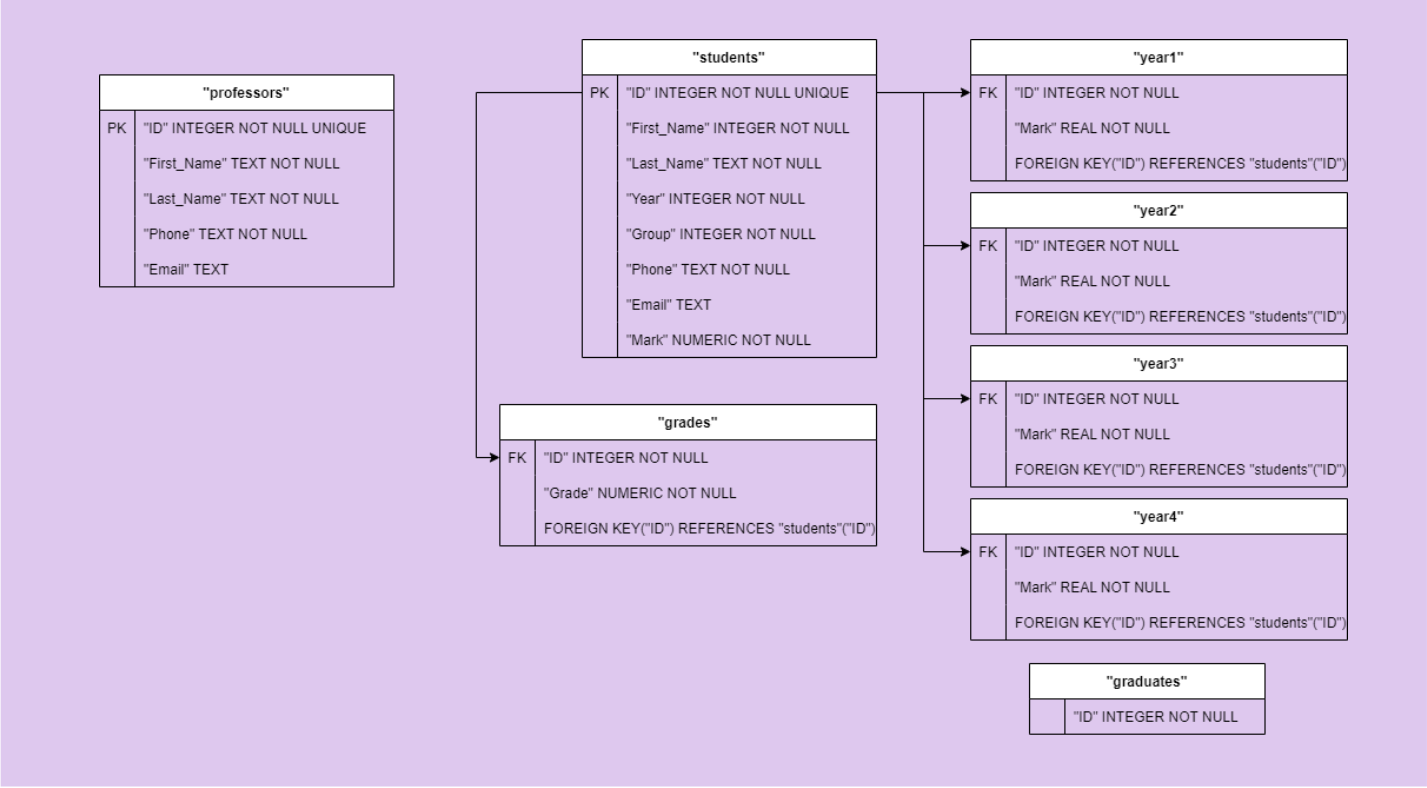
Scopul aplicatiei:

- Vizualizare rapida a studentilor si a mediilor acestora, acestea fiind grupati pe ani si grupe si avand informatii personale cum ar fi numar de telefon, email.
- Vizualizarea tuturor profesorilor, acestea avand mai multe metode de contact (numar telefon + adresa email oficiala)
- Fiecare student are mai multe note, astfel media se calculeaza in functie de acestea.
- Fiecare an (4 ani in total) are clasamentul propriu, unde se regasesc toti studentii
- Fiecare table poate fi sortat dupa nume, prenume, medie/note si an (doar unde este cazul), atat ascendent cat si descendent.
- Tabelul are si o functie de trecere de an, in care va actualiza automat toate tabele: studentii cu media peste 5 vor trece in anul urmator. In cazul in care anul current este ultimul, acestea vor fi eliminate din tabelul students si vor fi adaugati in tabelul graduates, in care se poate regasi numarul matricol (ID) al studentului. De asemenea, tabelele cu clasamentul pe fiecare an se va acualiza si acesta.
- Vizualizarea electronica este mult mai rapida decat modalitatile clasice, intrucat clasamentul se face mult mai usor, iar situatia studentilor se poate incheia cu o singura apasare de buton.

Tehnologii folosite

Aplicatia este realizata in **python** cu ajutorul librariei **tkinter** si **sqlite3**. Acestea se conecteaza la baza de date numita “data.db”, aceasta fiind facuta in **DB Browser (SQLite).** In acest fel crearea unui server local / conectare la un server hostat nu sunt necesare, intrucat toate informatiile sunt stocate local, accesate imediat (prin intermediul aplicatiei **PyCharm**)

Diagrama cu tabelele



Explicatii tabele

Students

ID – numarul matricol (PRIMARY KEY, UNIQUE – nu exista doi studenti cu acelasi numar matricol, AUTOINCREMENT – generarea numarului matricol se face automat)

First_Name, Last_Name – prenume, nume

Year – anul curent

Group – Grupa curenta

Phone – numarul de telefon

Email – adresa oficiala

Mark – Media Generala

Toate categoriile (in afara de email) au constrangerea NOT NULL, intrucat un student are nevoie de toate aceste informatii pentru a putea fi inregistrat

Grades

ID – numar matricol (FOREIGN KEY – conexiune one-to-many cu tabelul student, un student are mai multe note deci se regaseste acelasi ID de mai multe ori, ON DELETE CASCADE – pentru a se sterge odata ce studentul este sters)

Grade - nota

Professors

ID – numar de inregistrare (PRIMARY KEY, UNIQUE – nu exista doi profesori cu acelasi numar de inregistrare)

First_Name, Last_Name – prenume, nume

Phone – numarul de telefon

Email – adresa oficiala

Toate categoriile (in afara de email) au constrangerea NOT NULL, intrucat un profesor are nevoie de toate aceste informatii pentru a putea fi inregistrat

Year1, Year2, Year3, Year4

ID – numar matricol (FOREIGN KEY – conexiune one-to-one cu tabelul students, fiecare student se regaseste o singura data intr-un table in functie de an, ON DELETE CASCADE – pentru a se sterge odata ce studentul este sters)

Mark – Media generala

Toate categoriile au constrangerea NOT NULL, astfel nu se poate lasa liber

Graduates

ID – numar matricol absolvent (NOT NULL – nu poate exista fara acesta)

Functii importante

Conectarea la baza de date:

```
def __init__(self, master):
    self.tree = None
    self.master = master
    self.master.title("Faculty Management System")

    # Set the initial size of the window (width x height)
    self.master.geometry("1800x800")

    # Database initialization
    self.conn = sqlite3.connect("data.db")
    self.cursor = self.conn.cursor()

    # Create and set up GUI components
    self.create_widgets()
```

Functia pentru afisarea unui tabel(modular in functie de acesta):

```
students_button = tk.Button(frame1, text="Students", command=self.show_students_table)
students_button.pack(side="left", padx=5)
```

```
1 usage
def show_students_table(self):
    set_table(1)
    self.show_table("students")
```

```
def show_table(self, table_name):
    # Retrieve data from the selected table
    query = f'SELECT * FROM {table_name}'
    self.cursor.execute(query)
    table_data = self.cursor.fetchall()

    if hasattr(self, 'tree') and isinstance(self.tree, ttk.Treeview):
        self.tree.destroy()

    self.tree = ttk.Treeview(self.frame2, columns=self.get_column_names(table_name), show='headings',
                             height=20)

    column_widths = (200, 200, 200, 200, 200, 200, 200, 200)  # Adjust these values as needed

    for i, column in enumerate(self.get_column_names(table_name)):
        self.tree.heading(column, text=column)
        self.tree.column(column, width=column_widths[i], anchor="center")

    for row in table_data:
        self.tree.insert( parent: '', index: 'end', values=row)

    self.tree.pack(padx=10, pady=10)
```

Functia pentru afisarea unui tabel sortat (sistem de detectie al tabelului deschis, modular pentru tipul de sortare)

```
last_asc_button = tk.Button(frame1, text="LAST NAME ASC", command=lambda: self.show_sorted( criteria: "Last_Name",  typo: "ASC"))
last_asc_button.pack(side="left", padx=2)
last_desc_button = tk.Button(frame1, text="LAST NAME DESC", command=lambda: self.show_sorted( criteria: "Last_Name",  typo: "DESC"))
last_desc_button.pack(side="left", padx=2)
```

```
8 usages
def show_sorted(self, criteria, typo):

    match verify_table:
        case 1:
            table_n = "students"
        case 2:
            table_n = "grades"
            criteria = "Grade"
        case 3:
            table_n = "professors"
        case 4:
            table_n = "year1"
        case 5:
            table_n = "year2"
        case 6:
            table_n = "year3"
        case 7:
            table_n = "year4"
        case 8:
            table_n = "graduates"

    self.show_sorted_table( table_name: f"{table_n}",  criteria: f"{criteria}",  type: f"{typo}")
```

```
def show_sorted_table(self, table_name, criteria, type):

    query = f'SELECT * FROM {table_name} ORDER BY {criteria} {type}'
    self.cursor.execute(query)
    table_data = self.cursor.fetchall()

    if hasattr(self, 'tree') and isinstance(self.tree, ttk.Treeview):
        self.tree.destroy()

    self.tree = ttk.Treeview(self.frame2, columns=self.get_column_names(table_name), show='headings',
                             height=20)

    column_widths = (200, 200, 200, 200, 200, 200, 200, 200)  # Adjust these values as needed

    for i, column in enumerate(self.get_column_names(table_name)):
        self.tree.heading(column, text=column)
        self.tree.column(column, width=column_widths[i], anchor="center")

    for row in table_data:
        self.tree.insert( parent: '',  index: 'end', values=row)

    self.tree.pack(padx=10, pady=10)
```

Functia pentru promovarea studentilor

```
graduates_button = tk.Button(frame1, text="PASS 1 YEAR", command=self.promovate)
graduates_button.pack(side="left", padx=2)
```

```
def promovate(self):
    query = f'INSERT INTO graduates SELECT ID FROM students WHERE "Year" = 4 AND "Mark" >= 5;'
    self.cursor.execute(query)
    query = f'DELETE FROM students WHERE "Year" = 4 AND Mark >= 5;'
    self.cursor.execute(query)
    query = f'UPDATE students SET "Year" = 4 WHERE "Year" = 3 AND Mark >=5;'
    self.cursor.execute(query)
    query = f'UPDATE students SET "Year" = 3 WHERE "Year" = 2 AND Mark >=5;'
    self.cursor.execute(query)
    query = f'UPDATE students SET "Year" = 2 WHERE "Year" = 1 AND Mark >=5;'
    self.cursor.execute(query)
    query = f'DELETE FROM year1 WHERE "Mark" >= 5;'
    self.cursor.execute(query)
    query = f'DELETE FROM year2 WHERE "Mark" >= 5;'
    self.cursor.execute(query)
    query = f'INSERT INTO year2 ("ID", "Mark") SELECT "ID", "Mark" FROM students WHERE "Year" = 2 AND Mark >=5;'
    self.cursor.execute(query)
    query = f'DELETE FROM year3 WHERE "Mark" >= 5;'
    self.cursor.execute(query)
    query = f'INSERT INTO year3 ("ID", "Mark") SELECT "ID", "Mark" FROM students WHERE "Year" = 3 AND Mark >=5;'
    self.cursor.execute(query)
    query = f'DELETE FROM year4 WHERE "Mark" >= 5;'
    self.cursor.execute(query)
    query = f'INSERT INTO year4 ("ID", "Mark") SELECT "ID", "Mark" FROM students WHERE "Year" = 4 AND Mark >= 5;'
    self.cursor.execute(query)
```

Coming Soon

- Inserarea studentilor cu date si note
- Stergerea manuala a studentilor din tabele
- Inserarea profesorilor cu date
- Modificarea grupei unui student