

Aplicație pentru achiziționarea de jocuri

Petrișor Eduard-Gabriel

1311A

Coordonator: Mironeanu Cătălin

Facultatea de Automatică și Calculatoare

Cuprins:

- 1) Descrierea Proiectului
 - a) Scopul aplicației
- 2) Tehnologii Folosite
- 3) Structura bazei de date
 - a) Diagrama de relații
 - b) Constrângeri
 - c) Conectare
- 4) Capturi de Ecran si Exemple Cod

1) Descrierea Proiectului

a) Scopul aplicației

- i) Aplicația se încadrează în categoria de aplicații CRUD, scopul acesteia este permiterea consumatorilor să achiziționeze jocuri din cadrul unui magazin fizic folosind mediul online, dar și asistarea managerilor/angajaților în a ține evidența stocului și a prețurilor pentru jocurile disponibile.

Astfel un utilizator simplu poate să-și facă un cont în aplicație, să se logheze, să navigheze lista de jocuri disponibile și să le achiziționeze. În schimb un utilizator admin poate adăuga jocuri noi, edita datele jocurilor deja existente și șterge jocuri.

2) Tehnologii folosite

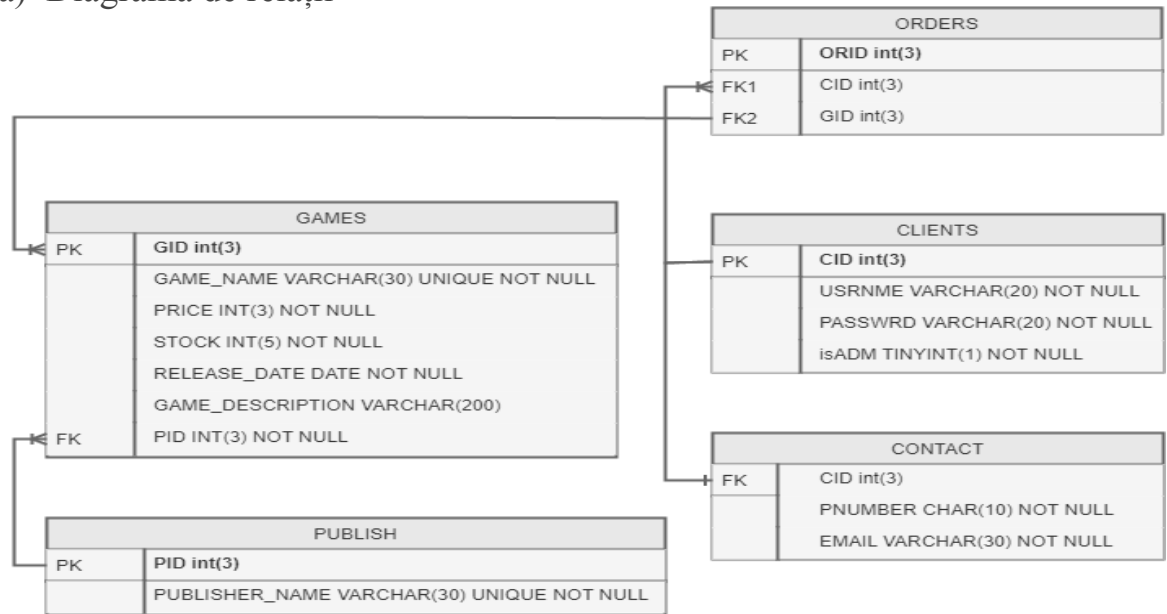
- a) În realizarea proiectului am folosit limbajul python atât pentru front-end cât și pentru back-end și MySQL pentru realizarea tabelelor din baza de date. Interfețele au fost realizate folosind framework-ul PyQt5.
- b) Conexiunea dintre back-end și baza de date MariaDB este realizată prin intermediul API-ului oferit de librăria Python mariadb

```
import mariadb
```

```
def connect():  
    try:  
        conn = mariadb.connect(  
            user="user",  
            password="user",  
            host="localhost",  
            database="TEMA_BD"  
        )  
    except mariadb.Error as e:  
        print(f"Error connecting to MariaDB Platform: {e}")  
        sys.exit(1)  
  
    return conn
```

3) Structura Bazei de Date

a) Diagrama de relații



b) Constrângeri folosite

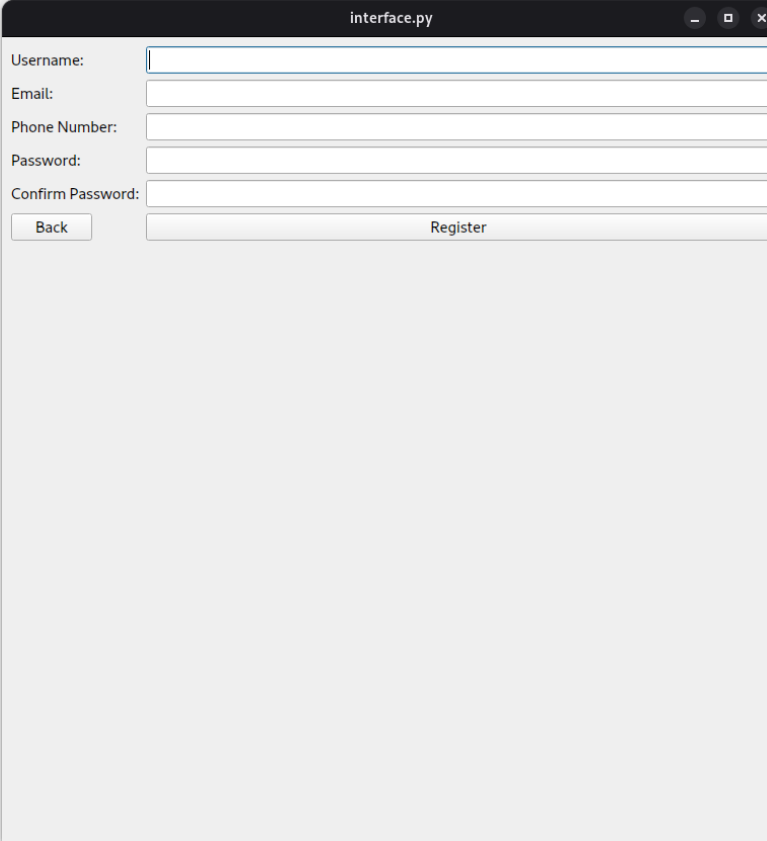
- PRIMARY KEY** - Cheile primare sunt prezente în majoritatea tabelelor și sunt generate prin funcționalitatea `AUTO_INCREMENT` din MySQL. Scopul acestora este să se asigure că fiecare rând dintr-o tabelă are un identificator unic.
- FOREIGN KEY** - Cheile străine sunt și ele prezente în majoritatea tabelelor și sunt folosite pentru asigurarea coerenței de date.
- NOT NULL** - Am folosit această constrângere pe majoritatea datelor deoarece în contextul aplicației nu ar fi avut sens ca acestea să fie de tip NULL.
- UNIQUE** - Această constrângere a fost folosită pentru asigurarea că date precum Numele de Jocuri, Numele de publishers, username-urile și emailurile nu se pot repeta.
- CHECK** - Au fost folosite pentru asigurarea că datele introduse în baza de date sunt în formatul corect, de exemplu un `CHECK CONSTRAINT` este folosit pentru asigurarea că numărul de telefon începe cu 07 și conține 10 caractere în total.

c) Conectare

- i) Aplicația se conectează automat la baza de date imediat după ce se deschide. Dacă se dorește conectarea la altă bază de date atunci trebuie editate datele de conectare prezente în apelul funcției connect din modulul mariadb, apelul acestei funcții se află în cadrul funcției connect din fișierul database.py.
- ii) Pentru generarea/umplerea/ștergerea/afișarea tabelor pot fi folosite scripturile prezente în fișierul scripts.

4) Capturi de Ecran si Exemple Cod:

a) Pagina de înregistrare



The screenshot shows a window titled "interface.py" with a registration form. The form contains five text input fields labeled "Username:", "Email:", "Phone Number:", "Password:", and "Confirm Password:". Below these fields are two buttons: "Back" on the left and "Register" on the right. The "Register" button is disabled, indicated by its lighter gray color.

Aceasta este funcția din front-end care preia datele introduse de user și le transmite către back-end.

```
def register_clicked(self):
    passwd = self.findChild(QLineEdit, 'password_edit').text()
    confpasswd = self.findChild(QLineEdit, 'confpassword_edit').text()
    username = self.findChild(QLineEdit, 'username_edit').text()
    pnumber = self.findChild(QLineEdit, 'pnumber_edit').text()
    email = self.findChild(QLineEdit, 'email_edit').text()

    if passwd == confpasswd and len(passwd) > 7:
        success, err_msg = new_user(self.parent().crs, username, email, pnumber, passwd)
        if success:
            self.reset_data()
            self.parent().conn.commit()
            self.parent().setCurrentIndex(2)
        else:
            QMessageBox.warning(self, 'Error', f'Failed to register user.\n {err_msg}')
    else:
        QMessageBox.warning(self, 'Error', 'Passwords do not match. Or Password too short')
```

Funcția din back-end vă verifică dacă datele sunt corecte și fie vă introduce datele în tabela, fie vă returna o eroare înapoi spre front-end în cazul în care exista o greșeală în datele introduse sau acestea încalcă un constraint. Eroarea apoi va fi afișată userului de către front-end.

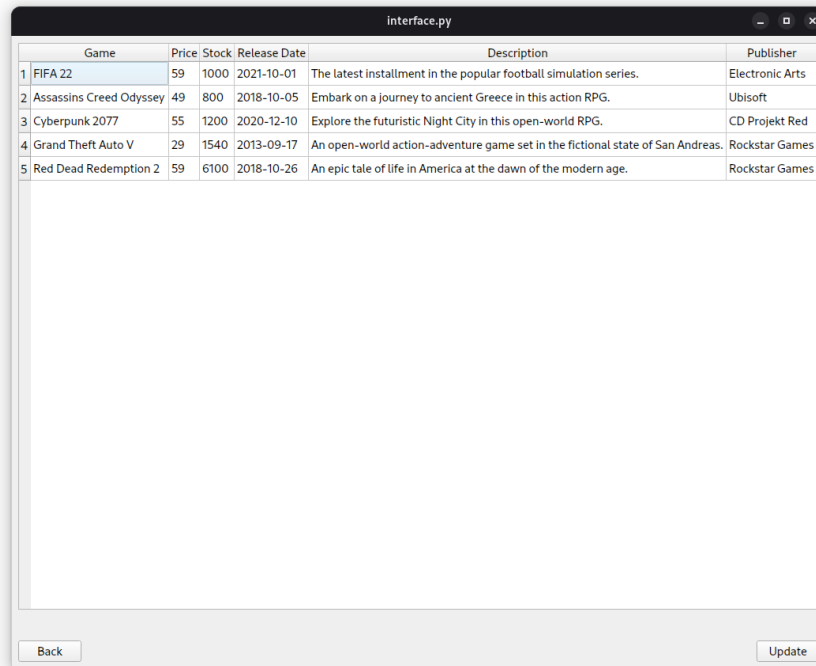
```
def new_user(cur, username, email, pnumber, passwd):
    try:
        cur.execute(
            "INSERT INTO CLIENTS (USRNME, PASSWRD) VALUES (?, ?)",
            (username, passwd)
        )

        cur.execute(
            "INSERT INTO CONTACT (CID, EMAIL, PNUMBER) VALUES (" +
            "(SELECT max(CID) FROM CLIENTS)" +
            ", ?, ?)",
            (email, pnumber)
        )

        return True, None
    except mariadb.Error as err:
        print(f"MySQL/MariaDB error: {err}")
        error_message = str(err)
        for code in err_codes.keys():
            if code in error_message[: len(error_message)//2]:
                return False, err_codes.get(code)

        return False, error_message
```

b) Pagina de Editare Jocuri a Adminului



	Game	Price	Stock	Release Date	Description	Publisher
1	FIFA 22	59	1000	2021-10-01	The latest installment in the popular football simulation series.	Electronic Arts
2	Assassins Creed Odyssey	49	800	2018-10-05	Embark on a journey to ancient Greece in this action RPG.	Ubisoft
3	Cyberpunk 2077	55	1200	2020-12-10	Explore the futuristic Night City in this open-world RPG.	CD Projekt Red
4	Grand Theft Auto V	29	1540	2013-09-17	An open-world action-adventure game set in the fictional state of San Andreas.	Rockstar Games
5	Red Dead Redemption 2	59	6100	2018-10-26	An epic tale of life in America at the dawn of the modern age.	Rockstar Games

Funcțiile `update_game_list()` și `populate_table()` sunt folosite pentru asigurarea că datele din tabela afișată sunt mereu up to date cu cele din baza de date.

```
def update_game_list(self):
    self.tableWidget.clearContents()
    self.populate_table()

def populate_table(self):
    data = get_games_data(self.parent().crs)
    self.gid = []
    self.data = []
    for g in data:
        self.gid.append(g[0])
    for temp in data:
        self.data.append(temp[1:])

    self.tableWidget.setRowCount(len(self.data))
    self.tableWidget.setColumnCount(len(self.data[0]))

    column_names = ["Game", "Price", "Stock", "Release Date", "Description", "Publisher"]
    self.tableWidget.setHorizontalHeaderLabels(column_names)

    for row in range(len(self.data)):
        for col in range(len(self.data[0])):
            item = QTableWidgetItem(str(self.data[row][col]))
            self.tableWidget.setItem(row, col, item)
    self.tableWidget.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)
```

Funcția `update_data()` este folosită pentru transmiterea modificărilor făcute de user către back-end unde acestea vor fi aplicate asupra bazei de date.

```
def update_data(self):
    updated_data = []
    for row in range(self.tableWidget.rowCount()):
        row_data = []
        for col in range(self.tableWidget.columnCount()):
            item = self.tableWidget.item(row, col)
            if item is not None:
                row_data.append(item.text())
            else:
                row_data.append("")
        updated_data.append(row_data)

    print("Updated Data:")
    for index, row in enumerate(updated_data):
        updated_data[index].append(self.gid[index])
        success, err_msg = update_games_data(self.parent().crs, updated_data[index])
        if success:
            self.parent().conn.commit()
        else:
            QMessageBox.warning(self, 'Error', f'Failed to update table.\n {err_msg}')
    self.parent().conn.commit()
```

```
def update_games_data(cur, data):
    try:
        cur.execute(
            """ Update GAMES SET
            GAME_NAME = ?,
            PRICE = ?,
            STOCK = ?,
            RELEASE_DATE = ?,
            GAME_DESCRIPTION = ?,
            PID = (SELECT PID FROM PUBLISH WHERE PUBLISHER_NAME = ?)
            WHERE GID = ? """ ,
            (data)
        )
        return True, None
    except mariadb.Error as err:
        print(f"MySQL/MariaDB error: {err}")
        error_message = str(err)
        for code in err_codes.keys():
            if code in error_message:
                return False, err_codes.get(code)
        return False, error_message
```