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

DISCIPLINA BAZE DE DATE

Gestiunea unei săli de jocuri video

Coordonator,
Cătălin Mironeanu

Student,
Iliescu Ștefan-Adrian
Grupa:1310A

Iaşi, 2022

Titlu proiect : Gestiunea activității unei săli de jocuri video

Analiza, proiectarea si implementarea unei baze de date care să modeleze activitatea unei săli de jocuri video.

Descrierea cerințelor și modul de organizare al proiectului

Pentru o bună activitate și organizare a unei săli de gaming, va trebui să implementăm o aplicație care să înregistreze și să afișeze informații despre clienții care au achitat sesiuni, și bineînțeles timpul jucat de client, în funcție de câți bani a dat.Fiecare joc aparține unui tip înregistrat în baza de date. Pentru a nu acumula un volum mare de informații, este bine să ștergem din jurnalul de achitări, înregistrări în care data curentă este mai mare față de data finalizării sesiunii înregistrării aferente.

Informațiile de care avem nevoie sunt cele legate de :

- **clienți(sau jucători):** datele referitoare la **clienți** se află în entitatea **Jucător,** dat fiind faptul că ne interesează numele acestuia, numărul cardului (0 dacă nu are), numele echipei din care face parte dacă are. Entitatea **Detalii_jucător** oferă mai multe detalii opționale precum e-mail-ul, genul și data_nașterii.

- **jocuri:** în cazul informațiilor despre **jocuri**,acestea se vor stoca in două entități astfel: într-o entitate părinte vom stoca informațiile cu privire la tipul de joc(**Tip_Joc**):ID-ul tipului și denumirea lui; în cealaltă entitate,cea de sesiune de gaming(**Sesiune_gaming**), vom stoca numele jocului. Din punct de vedere relațional, în tabela de sesiune de gaming va avea și ID-ul tipului de joc.

- **sesiuni:** în cazul informațiilor despre **sesiuni,** logic, în entitatea **Sesiune_gaming** se vor stoca ID-ul sesiunii și denumirea jocului.Relațional se vor stoca în tabelele **Sesiune_gaming** și **Achitare**,ID-ul sesiunii; iar doar în **Achitare**, data și ora achitării ei și cele ale finalizării.Cu aceste dăți, vom putea afla timpul jucat de client până momentul curent, și dacă s-a jucat mai mult decât trebuie.

- **plăți:** în cazul plăților, în tabela **Achitare**,pe lângă numărul cardului clientului și detalii despre sesiunea achitată menționată mai sus, se vor stoca și numărul bonului și prețul în funcție de câte ore vrea să joace clientul respectiv.

Aplicația dispune de 5 tab-uri ce conține câte un tabel pentru fiecare dintre tabelele bazei de date:**Jucător**,**Detalii Jucător**,**Tipuri de Joc**,**Sesiuni de Jocuri** și **Achitări**. La fiecare tab se poate apăsa click dreapta și se afișează 3 opțiuni de modificat tabela respectivă:*adăugare, modificare* și *ștergere*. Ultimele două opțiuni nu vor funcționa dacă nu s-a selectat un rând(va apărea avertisment).La adăugare și modificare se va afișa o fereastră pop-up cu casetele text sau combobox-uri(în cazul FOREIGN KEY-urilor sau în caz de avem voie de selectat numai una din puține valori) aferente atributelor tabelei respective.

Pe fereastra principală vom avea 3 butoane:*Commit,Rollback* și *Timp Jucat*.Primele două butoane sunt pentru tranzacție,auto-commit-ul fiind dezactivat,iar ultimul e pentru a afișa o fereastră cu o listă de informații despre timpul jucat de la început până acum de fiecare jucător.

Când apăsăm pe tab-ul de **Achitări** se va afișa un buton,*Curăță Achitările* care șterge din tabelă rândurile a căror timp de jucat a expirat.

Tehnologii utilizate

Aplicația este scrisă în *Java* folosind programul **IntelliJ IDEA**. Pentru partea de front-end s-a utilizat platforma *JavaFX* împreună cu aplicația Scene Builder pentru a crea interfața, aspectul acesteia fiind îmbunătățită prin CSS(Scene Builder dispune de mici opțiuni în niște câmpuri de text). Baza de date este creată și gestionată cu SQL Oracle, folosind **SQL Developer**. Prin driver-ul JDBC se va conecta aplicația la baza de date, deci trebuie descărcat pachetul ojdbc11.jar și trebuie inclus în structura proiectului(File ->Project Structure->Libraries->New Project Library->Java).

Descrierea funcțională a aplicației

Principalele funcţii care se pot întâlni într-o sală de jocuri video sunt:

- ✓ Înregistrarea și evidența clienţilor
- ✓ Evidenţa plăților efectuate
- ✓ Gestionarea timpului petrecut de către jucător

Descrierea detaliată a entităților si a relațiilor dintre tabele

Tabelele din aceasta aplicatie sunt:

- ✓ Detalii_jucator;
- ✓ Jucator;
- ✓ Achitare;
- ✓ Sesiune_gaming
- ✓ Tip_joc;

În proiectarea acestei baze de date s-au identificat tipurile de **relații** 1:n, 1:1, m:1 și m:n(doar în cazul logic).

Între tabelele **Detalii_Jucător** si **Jucător** se întâlnește o relație de tip **1:1,** fiind singura relație de acest tip din baza de date, deoarece un jucător poatea avea un singur set de detalii distincte, dar acel set aparține unui singur jucător. Legătura dintre cele două tabele este realizată prin câmpul *Nr_card***.**

Între **Tip_joc** și **Sesiune_gaming** se stabilește o legătură de tip one-to-many. Tabela **Tip_joc** descrie toate categoriile de jocuri ce le pune sala la dispoziția clienților, în timp ce tabela **Sesiune_gaming** prezintă clienților sesiunile de jocuri. Legătura dintre cele două tabele este făcută de câmpul *ID_tip***,** tabela părinte fiind **Tip_joc** deoarece pot fi mai multe sesiuni a căror joc este de un anume tip de joc , dar jocul unei sesiuni nu se poate încadra decât într-un singur tip.

Între entitățile **Jucător** și **Sesiune_gaming** este o legătură de tip many-to-many deoarece un jucător poate achita mai multe sesiuni pe numele lui, și o sesiune poate fi achitată de mai multe persoane. Pentru ca tabela să se afle in a 3-a formă normală, din punct de vedere relațional, această relație se va sparge în două, rezultând două relatii 1:m și m:1 și legătura între cele două tabele se va realiza cu ajutorul unei alte tabele **„Achitare”** care va conține cheia primară a fiecărei din cele două tabele. Atfel spus legătura se face cu ajutorul a două câmpuri: *Nr_card* și *ID_sesiune* reunite într-o tabelă comună,tabelele părinți fiind **Jucător** și **Sesiune_gaming**. Între **Jucător** și **Achitare** va fi relația 1:m pentru că un jucător poate face mai multe achitări, dar o achitare poate fi făcută doar de o singură persoană.Între **Achitare** și **Sesiune_gaming** este relația de tip m:1 fiindcă o achitare se axează pe o singură sesiune, dar sesiunea poate fi achitată de mai multe ori.

Descrierea constrângerilor

În tabela **Detalii_Jucător** este definită pe câmpul *Email* o constrângere de tip UNIQUE numită E-mail_UN(nu pot exista mai multe persoane care au aceeași adresă de mail),și în același timp o constrângere de tip CHECK numită email_CK care verifică adresa scrisă corect. Tot în entitatea aceasta sunt definite o constrângere gen_CK de tip CHECK care demonstrează că valoarea câmpului *Gen* este 'M' sau 'F', o constrângere birthdate_CK de tip CHECK care arată că valoarea câmpului *Data_nasterii* se află în intervalul de dăți 01-01-1950 și 01-01-2013. Pe câmpul *Nr_card* sunt definite două constrângeri: una de tip PRIMARY KEY numită Detalii_jucator_PK și una de tip FOREIGN KEY, Detalii_jucator_Jucator_FK cu care se realizează legătura între entitatea asta și entiatea **Jucător**.

În tabela **Jucător**, pe câmpul *Nr_card* este definită constrângerea Jucator_PK de tip PRIMARY KEY, iar pe câmpul Nume, constrângerea nume_CK care verifică dacă numele nu este doar un singur caracter și dacă nu există nicio cifră în nume.

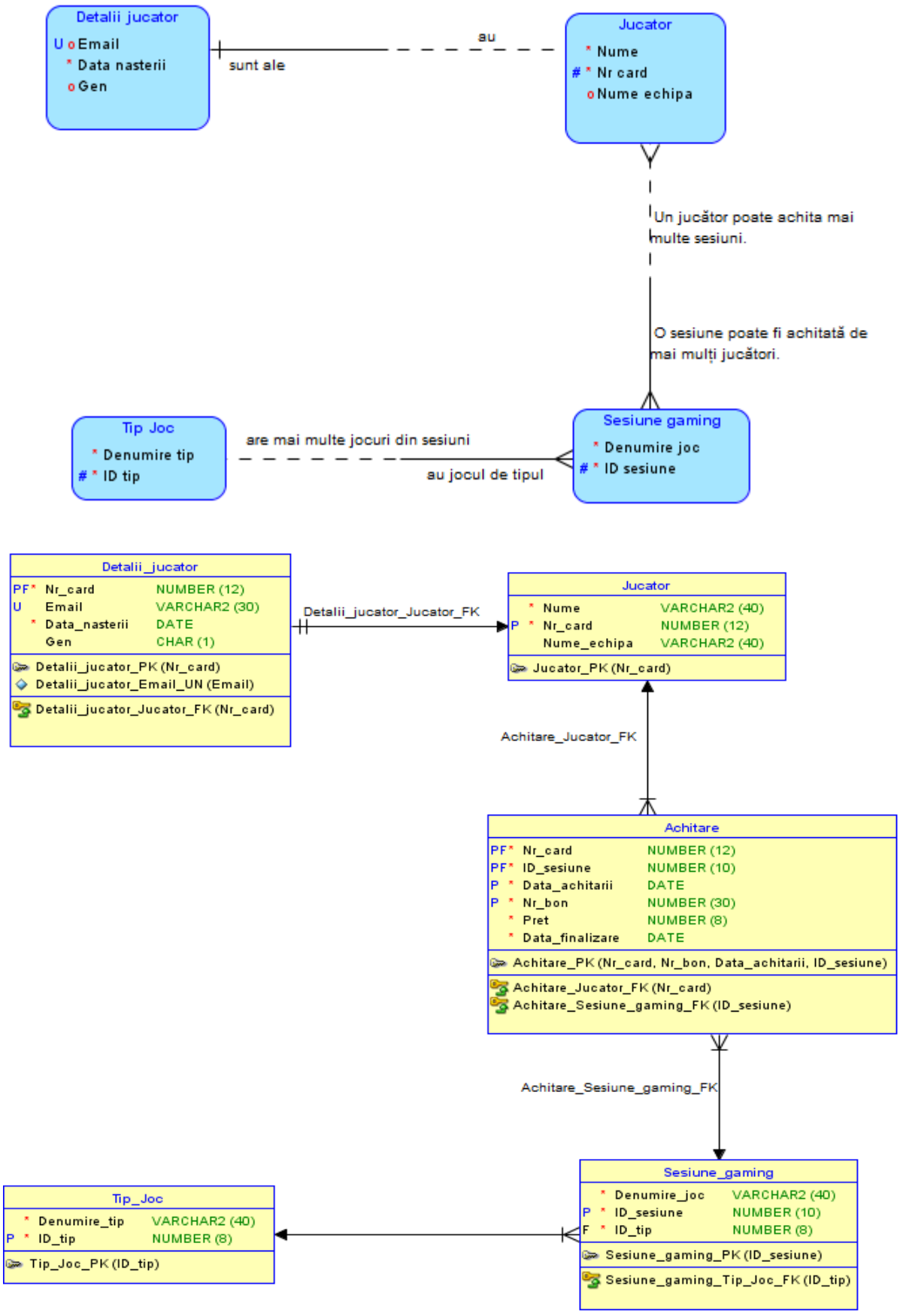
În tabela **Tip_Joc**, pe lângă constrângerea Tip_Joc_PK de tip PRIMARY KEY definită pe *ID_tip*, se va realiza și un auto-increment care se va începe de la 1 și se incrementează cu 1.

În tabela **Sesiune_gaming** se va face același lucru pe câmpul *ID_sesiune* precum în tabela **Tip_Joc**, numai că numele constrângerii este Sesiune_gaming_PK. Pe câmpul *ID_tip* este definită o constrângere de tip FOREIGN KEY: Tip_Joc_FK și cu aceasta se realizează legătura cu tabela menționată mai sus.

În tabela **Achitare**, constrângerea PRIMARY KEY,Achitare_PK este definită pe câmpurile *Nr_card*, *Nr_bon*,*Data_achitarii* și *ID_sesiune*. Pe câmpurile *Nr_card* și *ID_sesiune* sunt definite câte o constrângere de tip FOREIGN KEY: Achitare_Jucator_FK și Achitare_Sesiune_gaming_FK și cu acestea se realizează legături cu tabelele **Jucător** și **Sesiune_gaming**. Pe câmpul *Nr_bon* se face auto-incrementarea începând de la 100, adunând de fiecare dată cu 100. Constrângerea Pret_ck de tip CHECK verifică dacă *Pret* este 5 sau 10. Data_finalizare trebuie să fie egală cu *Data_achitarii*+*Pret*/120 folosind finish_ck de tip CHECK.

În cazul valorilor nule, singurele câmpuri care pot fi nule sunt *Email* și *Gen* din **Detalii_jucator** și *Nume_echipa* din **Jucator.**

Diagrama Entitate Relatie



Conectare la baza de date

Aplicația se conectează la baza de date prin intermediul JDBC-ului după ce se pornește. Mai întâi driver-ul se încarcă în aplicație, iar apoi, dacă detaliile de conectare sunt bune, se va conecta la server-ul facultății cu contul propriu de oracle pentru a accesa tabelele necesare.

```java
package com.example.gamingroom;
//Clasa abstracta ce are ca scop crearea conexiunii SQL statice si rularea unor comenzi SQL pentru fiecare tabela
import javafx.collections.ObservableList;
import java.sql.*;

public abstract class SQLConnection<T> {
    public static Connection conn;
    public static Statement stmt;

    public static void makeConnection() throws ClassNotFoundException, SQLException {
        //Incarcam driverul
        Class.forName("oracle.jdbc.driver.OracleDriver");
        //Cream obiectul conexiune
        conn = DriverManager.getConnection( url: "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl", user: "bd132", password: "bd132");
        conn.setAutoCommit(false);//pentru butoanele de commit si rollback
        stmt = conn.createStatement();

        System.out.println("Conectat!");
    }
    public static void closeConenction() throws SQLException {
        conn.close();
    }
    public static void commit() throws SQLException {
        conn.commit();
    }
    public static void rollback() throws SQLException {
        conn.rollback();
    }
```

Funcționalități

- **Select**: lista generată de funcție va fi inclusă ca parametru în funcția setItems a tabelului aferent. Exemplu din tabela **Detalii_jucător**:

```java
@Override
public ObservableList<DetaliiJucator> Select() throws SQLException {
    ObservableList<DetaliiJucator> list = FXCollections.observableArrayList();
    ResultSet rs=stmt.executeQuery( sql: "SELECT "+getColumns()+" FROM detalii_jucator");
    while(rs.next())
    {
        list.add(new DetaliiJucator(rs.getInt( columnIndex: 1),rs.getString( columnIndex: 2),rs.getDate( columnIndex: 3),rs.getString( columnIndex: 4)));
    }
    return list;
}
```

Sala de Gaming "Winners"

Jucător | Detalii Jucător | Tipuri de Joc | Sesiuni de Jocuri | Achitări | Timp jucat | Rollback | Commit

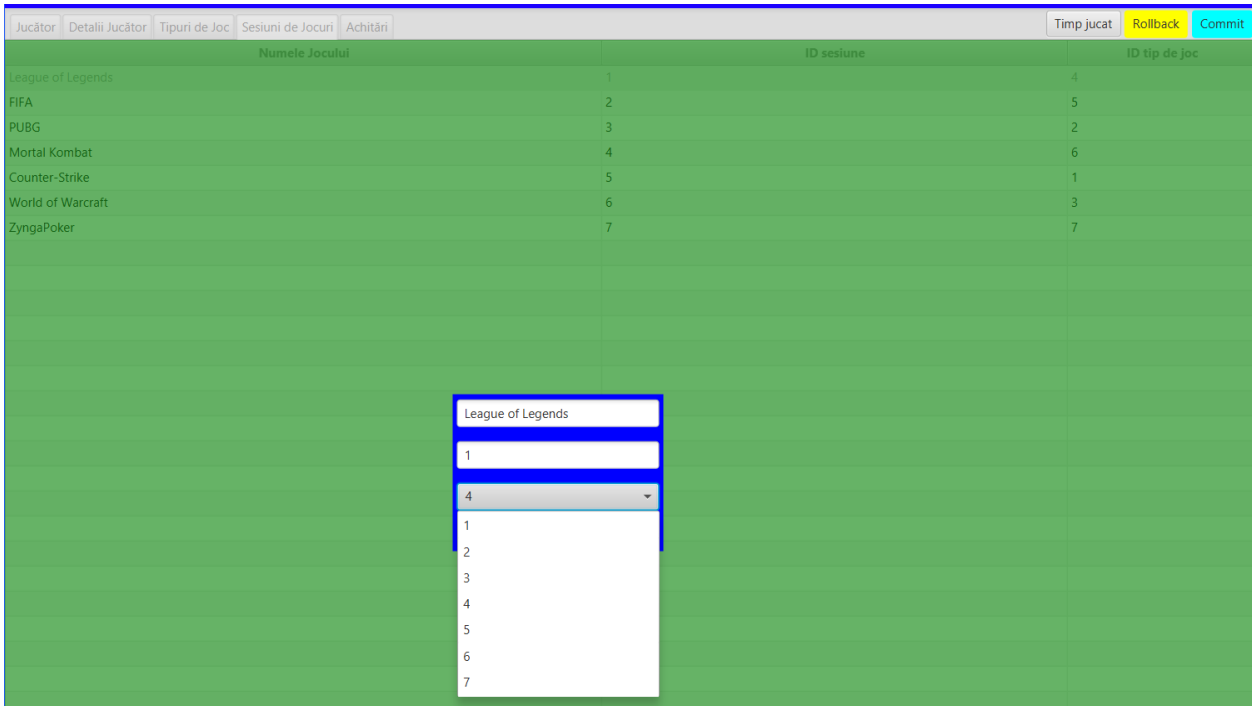
Număr card	Adresă de e-mail	Data nașterii	Gen
1782324569	faneiliescu@yahoo.com	2000-06-03	M
1986422137	emilut@gmail.com	2000-02-02	M
1430825739		2000-04-04	
1883267545	razvan_andrei@gmail.com	1999-10-29	M
1177809543	hellothere@yahoo.com	2001-12-10	F
1566980713	mm2001@outlook.com	2001-11-15	F
1779064352	george1996@msn.com	1995-11-15	M
1396407528	potop@conti.com	1980-08-18	M
1640193750	leo1990@superbet.ro	1990-06-22	M

- **Insert**: Exemplu de inserare în tabela **Jucător**:

```
public class JucatorSQL extends SQLConnection<Jucator> {
    @Override
    public String getColumns() { return "nume,nr_card,nume_echipa"; }
    @Override
    public void Insert(Jucator Obj) throws SQLException {
        String query = "INSERT INTO Jucator("+getColumns()+") VALUES('"+
                Obj.getNume()+"',"+Obj.getNr_card()+",";
        if(Obj.getNume_echipa()==null)
            query+="NULL)";
        else
            query+="'"+Obj.getNume_echipa()+"')";
        stmt.executeUpdate(query);
    }
```

- **Update:** Exemplu de update în tabela **Sesiune de Jocuri:**

```
@Override
public void Update(SesiuneJoc WhereObj, SesiuneJoc Obj) throws SQLException {
    String query="UPDATE sesiune_gaming SET denumire_joc='"+Obj.getDenumire_joc()+"', id_sesiune=";
    if(Obj.getId_sesiune()!=0)
        query+=Obj.getId_sesiune()+ ", id_tip="+Obj.getId_tip()+" WHERE id_sesiune="+WhereObj.getId_sesiune();
    else
        query+="SESIUNE_GAMING_ID_SESIUNE_SEQ.nextval, id_tip="+Obj.getId_tip()+" WHERE id_sesiune="+WhereObj.getId_sesiune();
    System.out.println(query);
    stmt.executeUpdate(query);
}
```



- **Delete:** Exemplu de ștergere în tabela **Achitare:**

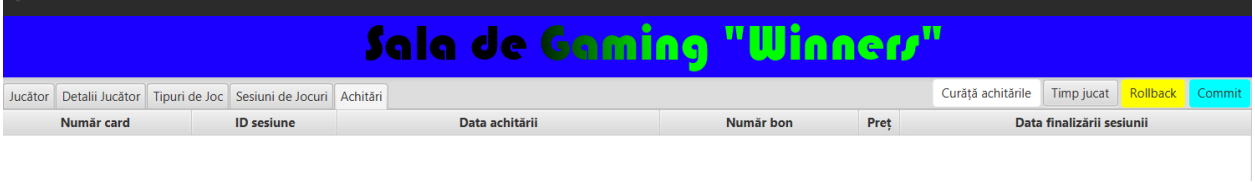
```
@Override
public void Delete(Achitare Obj) throws SQLException {
    String query="DELETE FROM achitare WHERE nr_bon="+Obj.getNr_bon();
    stmt.executeUpdate(query);
}
```

- **Afișare timp jucat:** se va afișa o listă cu timpul jucat pentru fiecare înregistrare din tabelul de **Achitări:**

```
public ObservableList<String> HoursPlayed() throws SQLException {
    ObservableList<String> list = FXCollections.observableArrayList();
    String query= "SELECT  nume||'  '||denumire_joc||'  '||TRUNC(minutes_ / 60) || ':' || TRUNC(MOD(minutes_,  60)) as Played " +
            "FROM (SELECT  (sysdate - data_achitarii) * 24 * 60 AS minutes_,nume,denumire_joc " +
            "FROM  achitare,jucator,sesiune_gaming " +
            "WHERE jucator.nr_card=achitare.nr_card and sesiune_gaming.id_sesiune=achitare.id_sesiune)";
    ResultSet rs=stmt.executeQuery(query);
    while(rs.next())
    {
        list.add(rs.getString( columnIndex: 1));
    }
    return list;
}
```

- **Ștergere înregistrări cu timp expirat:**

```
public void TimeExpiredDelete() throws SQLException {
    stmt.executeUpdate( sql: "DELETE FROM achitare WHERE SYSDATE> data_finalizare");
}
```

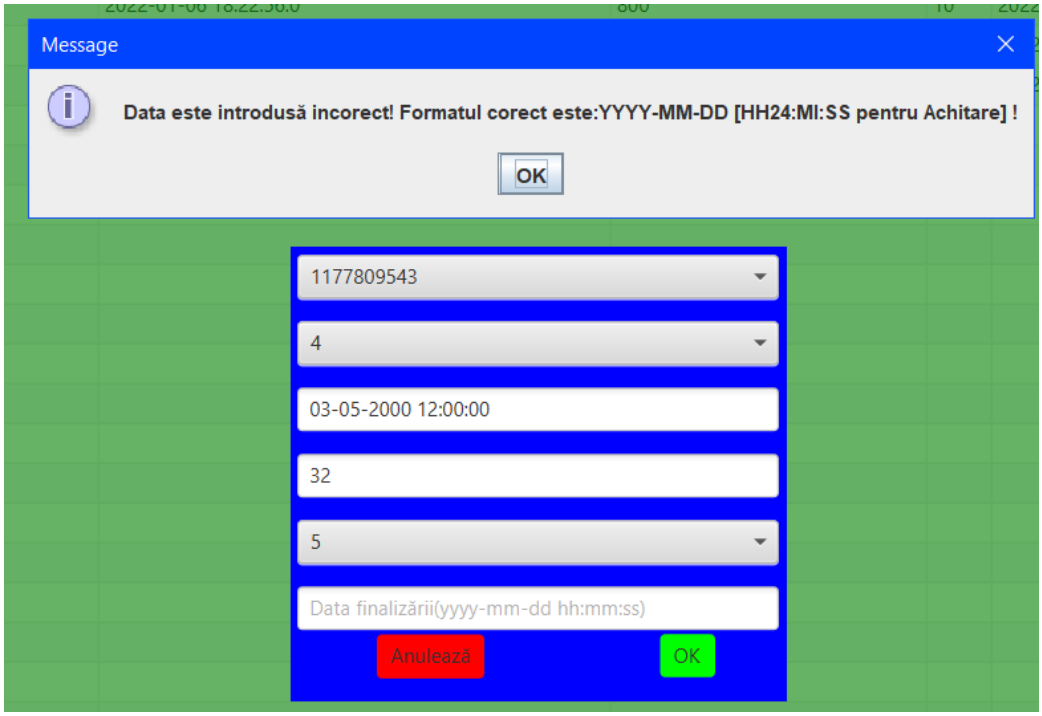


- **Tranzacție:** să presupunem că înainte de curățarea listei de achitări am făcut commit. Dacă facem rollback, vom avea valorile înapoi atunci când am dat commit.

Jucător | Detalii Jucător | Tipuri de Joc | Sesiuni de Jocuri | Achitări | Curăță achitările | Timp jucat | Rollback | Commit

Număr card	ID sesiune	Data achitării	Număr bon	Preț	Data finalizării sesiunii
0	3	2022-01-06 18:22:56.0	100	5	2022-01-06 19:22:56.0
1782324569	5	2022-01-06 18:22:56.0	200	10	2022-01-06 20:22:56.0
1986422137	5	2022-01-06 18:22:56.0	300	10	2022-01-06 20:22:56.0
1430825739	6	2022-01-06 18:22:56.0	400	5	2022-01-06 19:22:56.0
1883267545	7	2022-01-06 18:22:56.0	500	5	2022-01-06 19:22:56.0
1177809543	1	2022-01-06 18:22:56.0	600	5	2022-01-06 19:22:56.0
1566980713	4	2022-01-06 18:22:56.0	700	10	2022-01-06 20:22:56.0
1779064352	4	2022-01-06 18:22:56.0	800	10	2022-01-06 20:22:56.0
1396407528	2	2022-01-06 18:22:57.0	900	10	2022-01-06 20:22:57.0
1640193750	2	2022-01-06 18:22:57.0	1000	10	2022-01-06 20:22:57.0

- **Validări:** Când se introduc date ce nu respectă formatul impus sau au caractere nepermise, comanda nu va mai fi acceptată și va apărea o fereastră de alertă care precizează ce fel de eroare a fost.




```
catch (SQLException e)
{
    showMessageDialog( parentComponent: null, message: "A apărut o eroare la baza de date!" +
            " Uitati-vă pe SQL Developer și verificati datele!\nAtentie mare la constrângeri!!!");
    return;
}
catch(NumberFormatException e)
{
    showMessageDialog( parentComponent: null, message: "S-a introdus o valoare nulă sau incorectă pe o casetă de text unde trebuie număr.");
    return;
}
catch(IllegalArgumentException e)
{
    showMessageDialog( parentComponent: null, message: "Data este introdusă incorect!" +
            " Formatul corect este:YYYY-MM-DD [HH24:MI:SS pentru Achitare] !");
    return;
}
```