Universitate Tehnică “Gheorghe Asachi” Iași
Facultatea de Automatică și Calculatoarea
Calculatoare și Tehnologia Informației
2021-2022

Aplicație de urmărit și gestionat o ligă de fotbal

Temă de casă la disciplina
Baze de Date

Realizat de: Murariu-Tănăsache Iulian
Grupa: 1311A
Prof. Coordonator: Mironeanu Cătălin

Descriere proiect:

Proiectul constă într-o aplicație care urmărește evoluția unei ligi de fotbal. Este o aplicație pentru desktop prin care un utilizator va putea să fie la curent cu ultimele meciuri jucate și cu meciurile programate pe viitor. De asemenea, poate să vadă detalii si statisctici despre echipele ce participă, jucătorii acestora și stadioanele pe care se joacă meciurile.

Aplicația dispune de două moduri de vizualizare, una user și una admin. Prin modul user, utilizatorul este prezentat cu interfața principală a aplicației care dispune de 5 tab-uri: *Matches*, *Players*, *League*, *Team*, *Match*. Tab-ul *Matches* afișează toate meciurile ligii, cu posibilitatea de a filtra meciurile în funcție de un interval de date în care să fie programate. Apăsând click dreapta pe un meci apare opțiunea de a vizualiza statistici despre meciul respectiv, ceea ce trimite utilizatorul la tab-ul *Match*, unde poate vedea evenimentele din meci și cum s-a descurcat fiecare echipă și alte detalii.

Tab-ul *League* conține tabela ligii unde sunt afișate toate echipele, punctele și rezultatele lor. Click dreapta deschide un meniu pentru fiecare echipă prin care se poate trece la tab-ul *Team* pentru detalii despre echipa respectivă. Se pot vedea detalii și despre orice stadion, fie apăsând pe *View Stadion* in meniul pentru echipă din *League* sau apăsând click pe numele unui stadion. Tab-ul *Players* prezintă o tabelă cu toți jucătorii și statisticile lor de pe toată durata ligii. Și de aici se poate ajunge la tab-ul *Team* pentru a vedea detalii despre echipa vreunui jucător. Tab-urile *League* și *Players* au o bară de căutare pentru a găsi rapid orice înregistrare din tabelele din tab-ul repectiv.

Modul admin este folosit pentru a modifica baza de date a aplicației. Aici sunt prezente cele 5 tabele folosite și de aici pot fi modificate prin adăugarea, ștergerea sau modficarea unei înregistrări. Pentru a accesa meniul cu cele 3 opțiuni se apasă click dreapta pe tabelă sau pe o înregistrare. Pentru adăugare sau modificare va apărea un nou chenar unde să fie introduse datele necesare. Pentru cazul coloanelor care sunt foreign key, în loc de un text field este un dropdown meniu de unde sunt toate cheile care există deja pentru coloana referențiată. La tab-ul pentru tabela de echipe există o opțiune suplimentară *Register a new team*. Această opțiune pornește o tranzacție: Autocommit-ul este oprit; Apare meniul pentru a adăuga o echipă nouă, apoi se pot adăuga jucători pentru această echipă până se apasă pe butonul *Done adding players*; La final trebuie adăgat și un stadion nou pentru această echipă

și dacă totul a decurs fără erori se face commit și se încheie tranzacția. În caz de eroare sau dacă acțiunea este oprită prin apăsarea butonului *Cancel* se va efectua rollback înainte de a fi adăugată echipa.

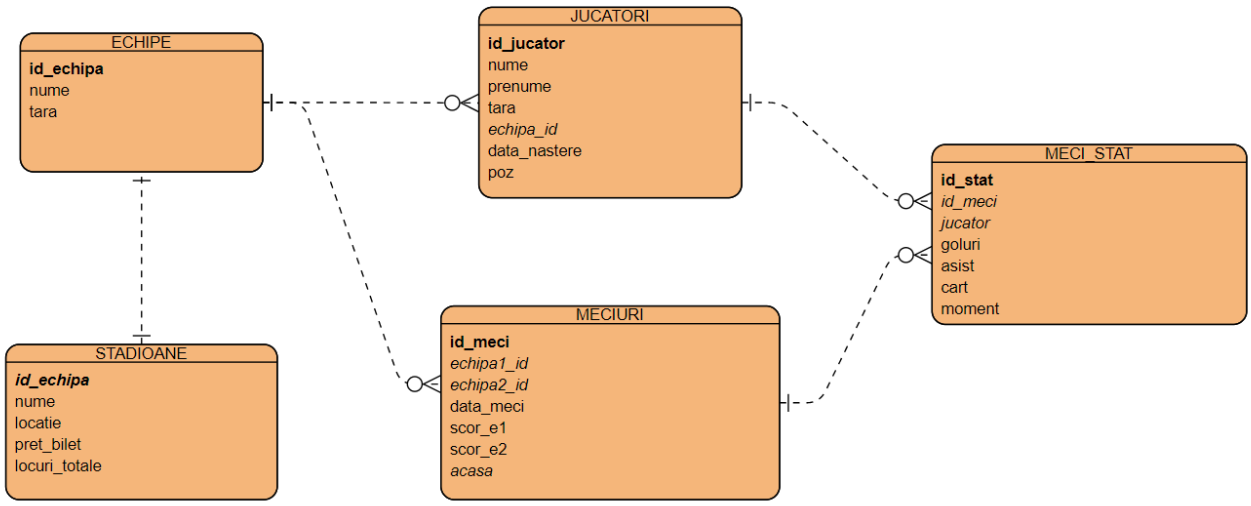
Tehnologii folosite:

Aplicația este scrisă în Java. Pentru partea de front-end am folosit platforma JavaFX împreună cu SceneBuilder pentru a crea interfața. De asemenea, am îmbunătățit aspectul interfeței utilizând CSS. Baza de date este creată și gestionată cu SQL Oracle, cu SQL Developer. Aplicația este conectată la baza de date prin driver-ul JDBC.

Baza de Date:

Baza de date din spatele aplicației cuprinde 5 tabele:

- *Echipe*: tabelă cu numele și țaraunei echipe, de această tabelă sunt legate multe relații cu alte tabele.
- *Jucatori*: tabelă ce are detalii despre jucători, cum ar fi nume, naționalitate, data nașterii, etc. Are un foreign key la tabela *Echipe* fiind o relație de one to many între jucători și echipe.
- *Meciuri*: tabelă ce are detalii despre meciuri, precum data, scorurile și echipele care joacă. Tabela are 3 foreign key-uri către tabela *Echipe* pentru a identifica cele 2 echipe care joacă și ce echipă joacă acasă.
- *Stadioane:* tabelă ce are detalii despre stadioane, cum ar fi număr de locuri, prețul unui bilet ș.a.m.d. Primary key-ul tabelei este id-ul echipei care joacă pe acel stadion, care este și foreign key, stabilindu-se o relație de one to one între echipă și stadion.
- *Meci_Stat:* tabelă ce are momente importante din meciuri. Are 2 foreign key-uri, unul către tabela *Meciuri* pentru a lega un meci de moment și unul către tabela *Jucatori* pentru a lega un jucător de acel moment, realizându-se o relație many to many. Tabela conține acțiunile unui jucător dintr-un meci, precum dacă a dat gol sau a luat un cartonaș și momentul când s-a întâmplat.



Constrângeri:

- **TABELA ECHIPE:**
  - id_echipa_pk – Constrângere Primary Key, reprezintă id-ul unei echipe
  - nume_echipa_nn – Constrângere Not Null, o echipă trebuie să aibă obligatoriu un nume
  - nume_echipa_uq – Constrângere Unique Key, numele unei echipe trebuie să fie unic.
  - nume_echipa_ck – Constrângere Check, numele unei echipe trebuie să fie mai lung de 2 caractere
  - tara_echipa_nn – Contrângere Not Null, o echipă trebuie să fie asociată cu o țară.
- **TABELA JUCATORI:**
  - id_jucator_pk – Constrângere Primary Key, reprezintă id-ul unui jucător
  - jucator_echipa_fk – Constrângere Foreign Key, leagă un jucător de o echipă
  - nume_check – Constrângere Check, verifică dacă numele unui jucător este mai lung de un caracter
  - nume_nn – Constrângere Not Null, un jucător trebuie să aibă un nume
  - prenume_nn, prenume_ck – Constrângeri identice cu cele de la nume, dar

pentru prenume.

- tara_jucator_nn – Constrângere Not Null**,** un jucător trebuie să aibă o naționalitate.

- **TABELA MECIURI:**
  - id_meci_pk – Constrângere Primary Key, reprezintă id-ul unui meci
  - meci_acasa_fk, meci_echipa1_fk, meci_echipa2_fk – Constrângeri Foreign Key, leagă echipele care joacă meciul și echipa care joacă acasă de tabela *ECHIPE*
- **TABELA STADIOANE:**
  - id_echipa_stadion_pk – Constrângere Primary Key, reprezintă id-ul echipei care joacă pe acest stadion
  - stadion_echipa_fk – Constrângere Foreign Key, leagă un stadion de o echipă
  - nume_stadion_nn – Constrângere Not Null, numele stadionului nu poate fi null
  - nume_stadion_uq – Constrângere Unique Key, numele stadionului trebuie să fie și unic
  - locatie_nn – Constrângere Not Null, un stadion trebuie să aibă o locație
  - pret_bilet_ck, locuri_totale_ck – Constrângeri Check, trebuie ca acestea să fie mai mari de 0, nu pot fi negative.
- **TABELA MECI_STAT:**
  - id_stat_pk – Constrângere Primary Key, reprezintă id-ul unui moment din tabelă
  - meci_stat_meci_fk, meci_stat_jucator_fk – Constrângeri Foreign Key, leagă un moment de un jucător și un meci
  - asist_ck, goluri_ck, cart_ck – Constrângeri Check, verifică ca aceste coloane să nu fie negative

Conectare la baza de date:

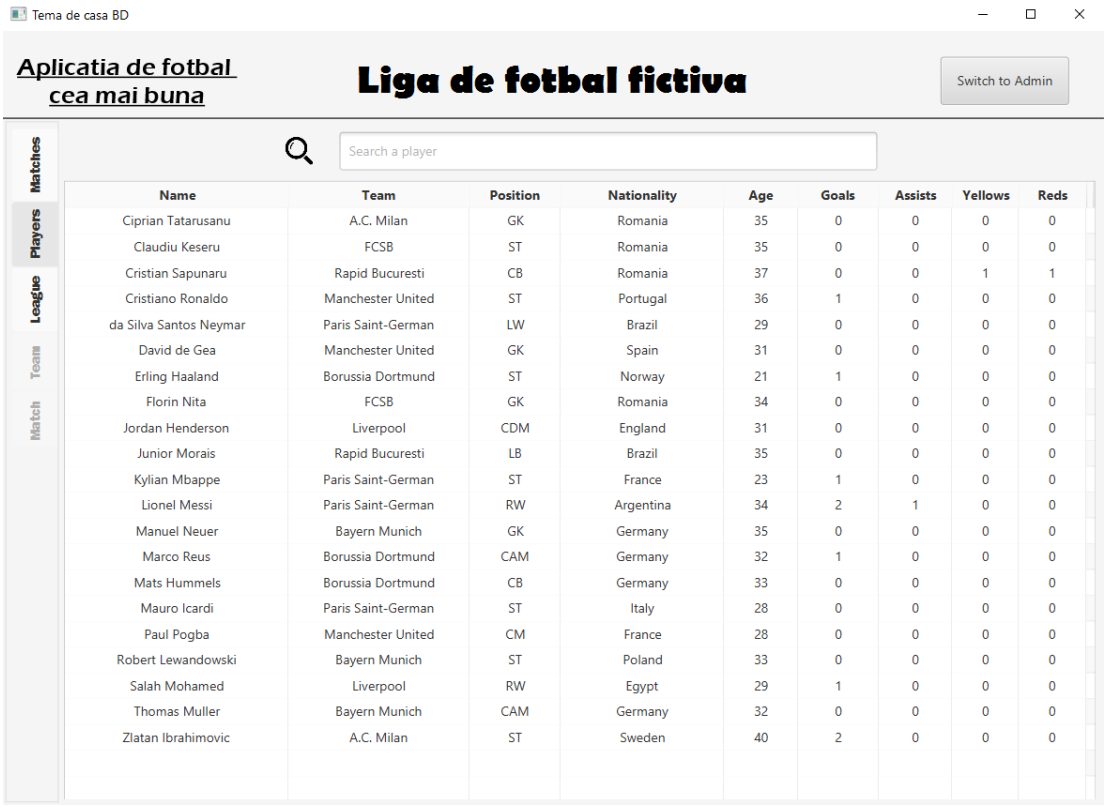
Aplicația se conectează la baza de date imediat când este pornită prin intermediul driver-ului JDBC. Acesta este încărcat în aplicație, iar apoi încearcă să se conecteze la server-ul facultății cu contul propriu de oracle pentru a accesa tabelele necesare.

```
//Clasa abstracta care se ocupa cu crearea unei conexiuni SQL, si comenzi specifice SQL pentru fiecare tabela
public abstract class SQLConnection<T> {

    protected static Connection con;
    protected static Statement stm;
    protected static PreparedStatement pstm;

    public static void makeSQLConnection() throws ClassNotFoundException, SQLException {
        //step1 load the driver class
        Class.forName("oracle.jdbc.driver.OracleDriver");

        //step2 create  the connection object
        con = DriverManager.getConnection( url: "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl", user: "bd161", password: "parola123");
        stm = con.createStatement();

        System.out.println("Connected!");
    }

    public static void closeSQLConenction() throws SQLException {
        con.close();
    }
```

Exemple de funcționalități:

- **Select**: Modul user permite doar vizualizarea datelor din baza de date prin diferite comenzi *Select.* De exemplu, tabela cu jucători din tab-ul *Players*:

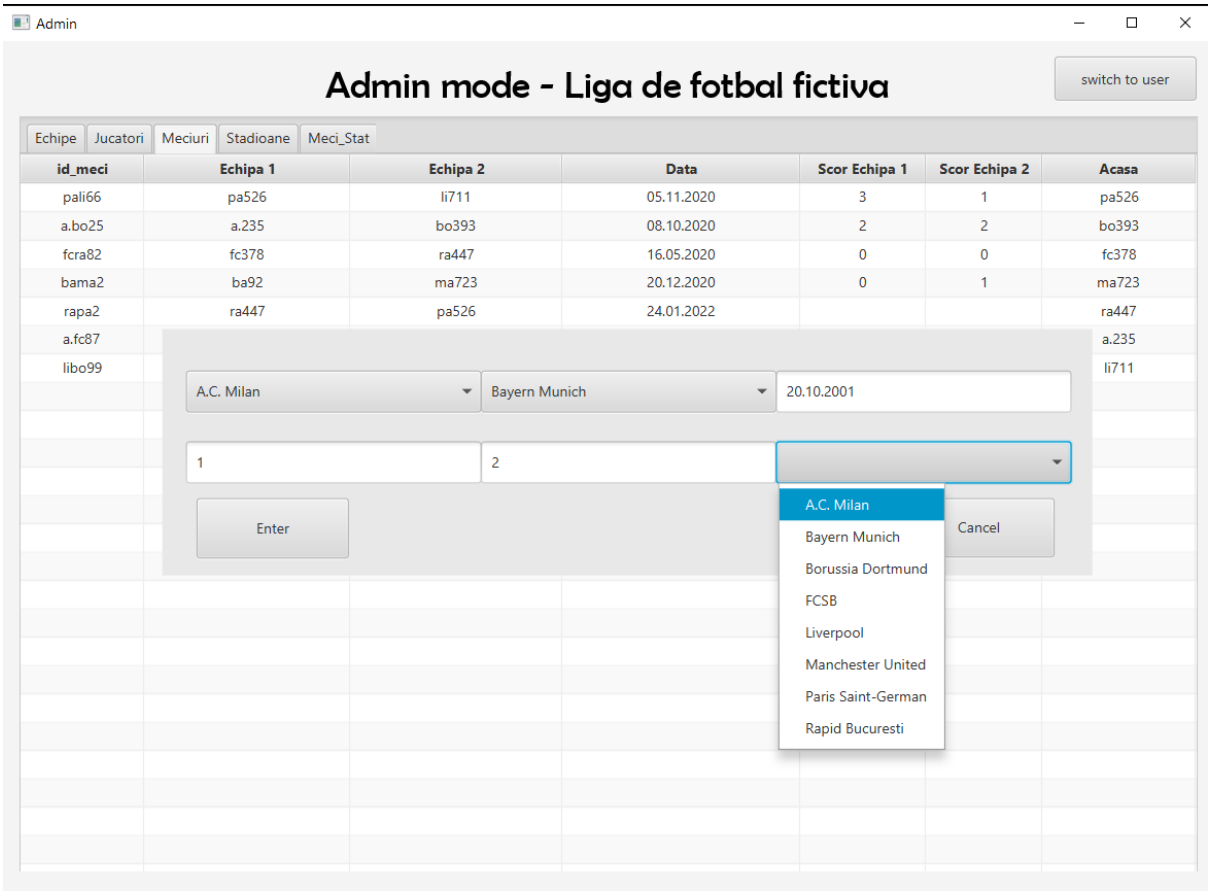
```
"SELECT DISTINCT jo.prenume || ' ' || jo.nume full_name, eo.nume, jo.poz, jo.tara, TRUNC((SYSDATE - jo.data_nastere)/365), gol, g.asist, galbene, rosii\n" +
"FROM JUCATORI jo, ECHIPE eo, MECI_STAT mso, \n" +
"    (SELECT id_jucator jucator, NVL(SUM(goluri),0) gol, NVL(SUM(asist),0) asist FROM MECI_STAT m, JUCATORI j WHERE m.jucator(+)=j.id_jucator GROUP BY id_jucator) g,\n" +
"    (SELECT id_jucator jucator, ss galbene FROM\n" +
"        (SELECT id_jucator, NVL(s,0) ss FROM JUCATORI, " +
"          (SELECT jucator ju, NVL(SUM(cart),0) s FROM MECI_STAT WHERE cart=1 GROUP BY jucator) WHERE id_jucator = ju(+))\n" +
"    )cg,\n" +
"    (SELECT jucator, ss rosii FROM\n" +
"        (SELECT id_jucator jucator, NVL(s,0) ss FROM JUCATORI, " +
"          (SELECT jucator j, NVL(SUM(cart),0)/2 s FROM MECI_STAT WHERE cart=2 GROUP BY jucator) WHERE id_jucator = j(+))\n" +
"    )cr\n" +
"WHERE jo.id_jucator=mso.jucator(+) AND\n" +
"      eo.id_echipa=jo.echipa_id AND\n" +
"      g.jucator=jo.id_jucator AND\n" +
"      cg.jucator=jo.id_jucator AND\n" +
"      cr.jucator=jo.id_jucator\n" +
"ORDER BY full_name";
```



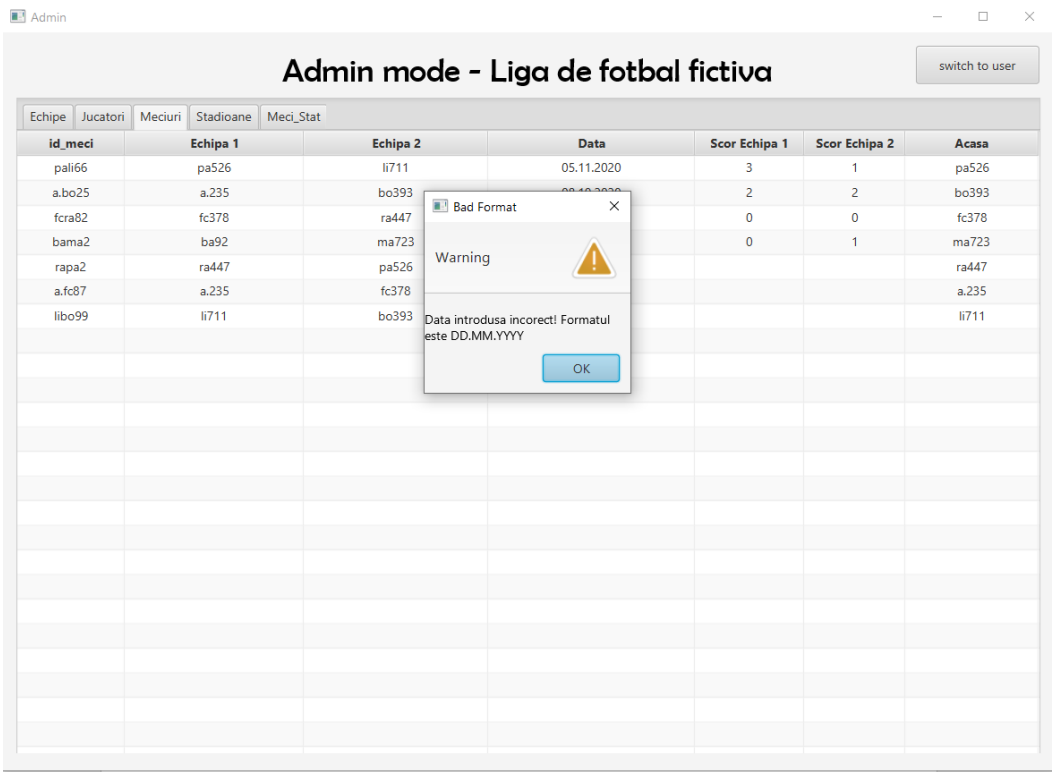
Name	Team	Position	Nationality	Age	Goals	Assists	Yellows	Reds
Ciprian Tatarusanu	A.C. Milan	GK	Romania	35	0	0	0	0
Claudiu Keseru	FCSB	ST	Romania	35	0	0	0	0
Cristian Sapunaru	Rapid Bucuresti	CB	Romania	37	0	0	1	1
Cristiano Ronaldo	Manchester United	ST	Portugal	36	1	0	0	0
da Silva Santos Neymar	Paris Saint-German	LW	Brazil	29	0	0	0	0
David de Gea	Manchester United	GK	Spain	31	0	0	0	0
Erling Haaland	Borussia Dortmund	ST	Norway	21	1	0	0	0
Florin Nita	FCSB	GK	Romania	34	0	0	0	0
Jordan Henderson	Liverpool	CDM	England	31	0	0	0	0
Junior Morais	Rapid Bucuresti	LB	Brazil	35	0	0	0	0
Kylian Mbappe	Paris Saint-German	ST	France	23	1	0	0	0
Lionel Messi	Paris Saint-German	RW	Argentina	34	2	1	0	0
Manuel Neuer	Bayern Munich	GK	Germany	35	0	0	0	0
Marco Reus	Borussia Dortmund	CAM	Germany	32	1	0	0	0
Mats Hummels	Borussia Dortmund	CB	Germany	33	0	0	0	0
Mauro Icardi	Paris Saint-German	ST	Italy	28	0	0	0	0
Paul Pogba	Manchester United	CM	France	28	0	0	0	0
Robert Lewandowski	Bayern Munich	ST	Poland	33	0	0	0	0
Salah Mohamed	Liverpool	RW	Egypt	29	1	0	0	0
Thomas Muller	Bayern Munich	CAM	Germany	32	0	0	0	0
Zlatan Ibrahimovic	A.C. Milan	ST	Sweden	40	2	0	0	0

- **Insert:** Se poate realiza doar din modul admin. Exemplu de introducere în tabela *Meciuri*:

```
//Implementarea clasei abstracte SQLConnection pentru tabela MECIURI
public class MeciSQL extends SQLConnection<Meci>{

    @Override
    public String getColumns() { return "id_meci,echipa1_id,echipa2_id,data_meci,scor_e1,scor_e2,acasa"; }

    @Override
    public void Insert(Meci last) throws SQLException {
        //insert ultima
        pstm = con.prepareStatement( sql: "INSERT INTO MECIURI(" + getColumns() + ") VALUES (?, (SELECT id_echipa FROM ECHIPE WHERE nume=?), (SELECT id_echipa FROM ECHIPE WHERE nume=?), TO_DATE(?,'DD
        pstm.setString( parameterIndex: 1, last.getId_meci());
        pstm.setString( parameterIndex: 2, last.getEchipa1_id());
        pstm.setString( parameterIndex: 3, last.getEchipa2_id());
        pstm.setString( parameterIndex: 4, last.getData_meci());
        pstm.setObject( parameterIndex: 5, last.getScor_e1(), java.sql.Types.INTEGER);
        pstm.setObject( parameterIndex: 6, last.getScor_e2(), java.sql.Types.INTEGER);
        pstm.setString( parameterIndex: 7, last.getAcasa());

        pstm.executeUpdate();
    }

    @Override
    public void Update(Meci toUpdate, Meci newItem) throws SQLException {
        pstm = con.prepareStatement( sql: "UPDATE MECIURI echipa1_id=(SELECT id_echipa FROM ECHIPE WHERE nume=?), echipa2_id=(SELECT id_echipa FROM ECHIPE WHERE nume=?), data_meci=TO_DATE(?,'DD.MM.YY

        //pstm.setString(1, newItem.getId_meci());
        pstm.setString( parameterIndex: 1, newItem.getEchipa1_id());
        pstm.setString( parameterIndex: 2, newItem.getEchipa2_id());
        pstm.setDate( parameterIndex: 3, Date.valueOf(newItem.getData_meci()));
        pstm.setObject( parameterIndex: 4, newItem.getScor_e1(), java.sql.Types.INTEGER);
        pstm.setObject( parameterIndex: 5, newItem.getScor_e2(), java.sql.Types.INTEGER);
        pstm.setString( parameterIndex: 6, newItem.getAcasa());
        pstm.setString( parameterIndex: 7, toUpdate.getId_meci());

        pstm.executeUpdate();
```



- **Validări:** În cazul în care au fost introduse date ce nu respectă formatul impus sau au caractere nepermise, comanda va fi invalidată și va apărea o aletă ce precizează eroarea.




```java
//Functie pentru creare alerta
public void Alerta(String title, String content, Exception e) {
    PopupMenu( visible: false);
    //Alerta utilizatorul de eroare
    Alert badType = new Alert(Alert.AlertType.WARNING);
    badType.setTitle(title);
    String msg = content + '\n' + (e != null ? e.getMessage() : "");
    Text label = new Text(msg);
    label.setWrappingWidth(200);
    if (e != null) {
        System.out.println(e.getMessage());
    }
    badType.getDialogPane().setContent(label);
    badType.show();
}

//Functie pentru butonul de enter pentru adaugarea/modificarea in tabel
public void OnEnter() {
    String[] param = new String[fields.size() + dropdowns.size()]; // parametrii noii inregistrari
    for (int i = 0; i < fields.size(); ++i) {
        param[i] = fields.get(i).getText();
    }

    for (int i = fields.size(); i < fields.size() + dropdowns.size(); ++i) {
        param[i] = dropdowns.get(i - fields.size()).getValue();
    }
    //Adaugarea unui element nou
    if (toAdd) {
        try {
            currTable.getSql().Insert(currTable.getAdaptor().createObject(param));
            currTable.setList(currTable.getSql().Select());
            currTable.getTable().setItems(currTable.getList());
        } catch (SQLException e) {
            Alerta( title: "SQL Error", content: "A aparut o eroare la baza de date! ", e);
        } catch (NumberFormatException e) {
```