



**Universitatea POLITEHNICA din București**

Facultatea de electronică, Telecomunicații și Tehnologia Informației



# **PROGRAMAREA INTERFEȚELOR PENTRU BAZE DE DATE Proiect tehnologia JavaFX**

**Profesor coordonator: Ș. I. Dr. Ing. Pupezescu Valentin**

**STUDENT: Stan Cătălin-Ionel**

**GRUPA: 432C**

## CUPRINS

1) Tema proiectului.....	pag. 3
2) Descrierea sistemului de gestiune a bazelor de date MySQL....	pag. 3
1. Ce este MySQL?.....	pag. 3
2. Cum administrează MySQL bazele de date?.....	pag. 4
3. Care sunt avantajele utilizării MySQL?.....	pag. 4
3) Tehnologia JavaFX.....	pag. 4
1. Ce este JavaFX?.....	pag. 4
2. Care sunt avantajele utilizării JavaFX?.....	pag. 4
4) Descrierea aplicației.....	pag. 5
1. Baza de date.....	pag. 5
2. Diagrama logică a bazei de date (Diagrama ERD).....	pag. 7
3. Funcționalitatea aplicației.....	pag. 8
1. Arhitectura proiectului.....	pag. 8
2. Implementarea funcțiilor.....	pag. 8
5) Concluzii.....	pag. 17
6) Bibliografie.....	pag. 17

## 1) Tema proiectului

Tema proiectului se bazează pe dezvoltarea unei aplicații ce conține o bază de date creată în sistemul de gestiune a bazelor de date MySQL. Asocierea dintre tabelele bazei de date va fi M:N.

Pentru crearea interfeței ne vom folosi de tehnologia JavaFX. Interfața va trebui să permită utilizatorului să execute pe toate tabelele bazei de date, următoarele operații: vizualizare, modificare, adăugare și ștergere, iar vizualizarea tabelului de legătură va presupune afișarea datelor referite din celelalte tabele.

## 2) Descrierea sistemului de gestiune a bazelor de date MySQL

### 2.1. Ce este MySQL?

MySQL este un sistem de gestiune a bazelor de date relaționale open source care este utilizat în principal pentru aplicațiile online. MySQL poate crea și gestiona baze de date foarte utile (cum ar fi informații despre angajați, inventar și multe altele), la fel ca alte sisteme, cum ar fi popularul Microsoft Access. În timp ce Microsoft Access, MySQL și alte sisteme de gestiune a bazelor de date servesc scopuri similare (de a găzdui datele), utilizarea diferă foarte mult. [1]

MySQL este componenta integrată a platformelor LAMP sau WAMP (Linux/Windows- Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului care este adesea combinat cu MySQL și denumit Duo-ul Dinamic. În multe cărți de specialitate este precizat faptul ca MySQL este mult mai ușor de învățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”. [1]

## **2.2. Cum administrează MySQL bazele de date?**

Pentru administrarea bazelor de date, MySQL oferă utilizatorului posibilitatea de a utiliza o interfață grafică (MySQL Workbench) sau linia de comandă. [1]

## **2.3. Care sunt avantajele utilizării MySQL?**

MySQL este compatibil cu o varietate de limbaje de programare, inclusiv PHP, Python, Java, C#, și altele. Acest lucru face integrarea cu aplicații software diverse mai ușoară. De asemenea, un alt avantaj al utilizării tehnologiei MySQL este reprezentat de suportul pentru tranzacții, lucru care ne permite să efectuăm operațiuni multiple asupra bazei de date într-un mod atomic (totul sau nimic).

MySQL oferă opțiuni avansate pentru replicare, permițând crearea de copii ale bazei de date pentru asigurarea redundanței și îmbunătățirea scalabilității. [1]

## **3) Tehnologia JavaFX**

### **3.1. Ce este JavaFX?**

JavaFX este o tehnologie de dezvoltare a interfețelor grafice pentru aplicații desktop și mobile în limbajul de programare Java. Aceasta oferă un set bogat de biblioteci și instrumente pentru a crea interfețe moderne, interactive și estetice. JavaFX permite dezvoltatorilor să creeze aplicații cu aspect profesional, cu funcționalități avansate precum animații, grafică 2D și 3D, efecte vizuale și interacțiuni tactilă.

Prin intermediul JavaFX, dezvoltatorii pot proiecta interfețe de utilizator flexibile, personalizabile și ușor de gestionat. Tehnologia este integrată strâns cu limbajul Java, beneficiind de toate avantajele acestuia, cum ar fi portabilitatea, securitatea și performanța. JavaFX este folosită pe scară largă în industrie pentru dezvoltarea de aplicații desktop moderne, dar poate fi adaptată și pentru dispozitive mobile și alte platforme, făcând-o o alegere versatilă pentru dezvoltatorii Java.[2]

### 3.2. Care sunt avantajele utilizării JavaFX?

- Oferă o performanță bună și scalabilitate, ceea ce înseamnă că poate gestiona cu ușurință aplicații de dimensiuni mari și complexe.
- Oferă facilități îmbunătățite pentru manipularea și afișarea conținutului multimedia, precum audio și video, făcându-l potrivit pentru aplicații care necesită gestionarea acestor elemente.
- Utilizează FXML, un limbaj de markup utilizat pentru definirea interfețelor grafice, și Scene Builder, o unealtă grafică pentru construirea acestor interfețe.
- Are capacitatea de a integra conținut web prin intermediul componentelor WebView, permițând încorporarea paginilor web în aplicații desktop.
- Beneficiază de o comunitate activă de dezvoltatori și suport din partea Oracle. Există o gamă largă de resurse, documentație și tutoriale disponibile pentru a ajuta dezvoltatorii să își îmbunătățească cunoștințele și să depășească eventualele probleme. [3]

## 4) Descrierea aplicației

### 4.1. Baza de date

Tema individuală se bazează pe crearea unei baze de date ce are 2 tabele în asociere M:N. Tabelele sunt: *abonati* și *companie\_telefonie*. Pentru cele 2 tabele am ales câteva atribute caracteristice:

Pentru tabela *abonati*, am ales ca și cheie primară *idAbonat*. Celelalte atribute sunt: *nume*, *prenume*, *cnp*, *adresa*.

Table Name: abonati

Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idAbonat	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
prenume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
cnp	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
adresa	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Pentru tabela **companie\_telefonie**, am ales ca și cheie primară **idCompanieTelefonie**. Celelalte atribute sunt: **nume**, **adresaSediu**, **telefonContact**, **emailContact**.

Table Name: companie\_telefonie

Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idCompanieTelefonie	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
adresaSediu	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
telefonContact	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
emailContact	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

## Ce este asocierea M:N?

Asocierea M:N (mai-mulți-la-mai-mulți) are ca și caracteristică faptul că fiecărui element înregistrat într-o tabelă îi pot fi asociate mai multe elemente din cealaltă tabelă și invers.

De exemplu, în cazul nostru, un abonat poate fi asociat mai multor companii de telefonie, așa cum și unei companii de telefonie îi pot fi asociați mai mulți abonați. [4]

## Ce reprezintă tabela de legătură?

Pentru a crea o relație mai-mulți-la-mai-mulți, trebuie să se creeze o a treia tabelă denumită deseori tabelă de joncțiune, care împarte relația mai-mulți-la-mai-mulți în două relații unu-la-mai-mulți. În cazul nostru, am ales ca și tabelă de legătură tabela abonamente.

În această nouă tabelă, atributele ce au fost selectate ca si chei primare pentru tabelele anterioare vor deveni chei străine (FK) pentru tabela de legătura abonamente.

Table Name: abonamente

Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
fk1	abonamentedb`.`abonati`		
fk2	abonamentedb`.`companie_telefonie`		

Pentru această tabelă, am ales atributul *idAbonament* ca și cheie primară. Restul atributelor sunt cheile străine *idAbonat* și *idCompanieTelefonie*.

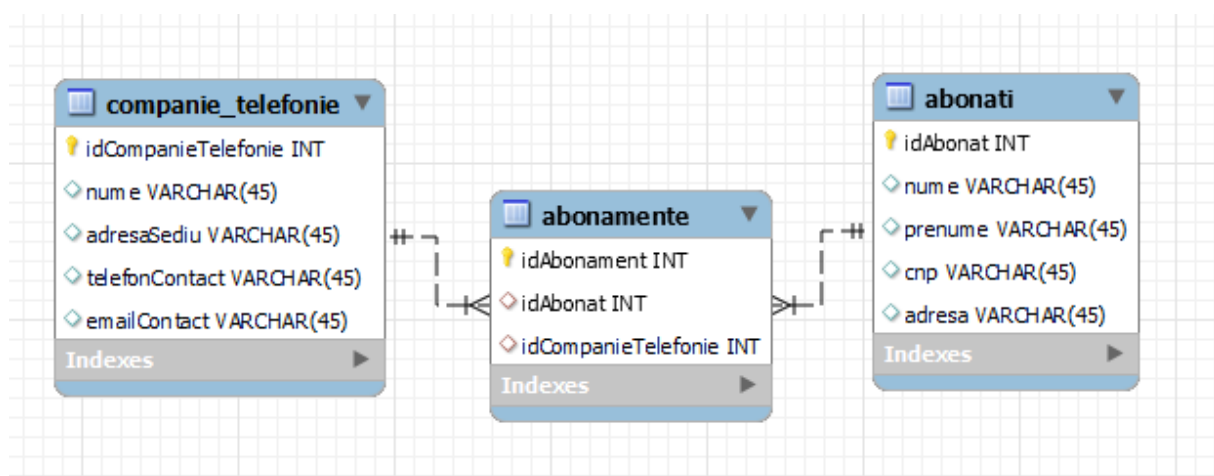
Table Name: abonamente

Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idAbonament	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
idAbonat	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
idCompanieTelefonie	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

## 4.2. Diagrama logică a bazei de date (Diagrama ERD)



### De ce este importanta diagrama logica a bazei de date?

Aceasta oferă o reprezentare logică detaliată a datelor celor 3 tabele, a relațiilor dintre ele.

### Ce relații există între cele trei tabele?

- Între *companie\_telefonie* și *abonamente* există o asociere de tip 1:N
- Între *companie\_telefonie* și *abonati* există o asociere de tip M:N
- Între *abonati* și *abonamente* există o asociere de tip 1:N

## 4.3. Funcționalitatea aplicației

### 4.3.1. Arhitectura proiectului

Proiectul realizat în tehnologia JavaFX are următoarea structură:

- Fișierul *DBOperations.java*, ce reprezintă “motorul” care realizează operațiile pentru conectarea bazei de date și gestionarea tabelelor și a datelor. Acesta conține toate metodele necesare implementării operațiilor CRUD ce se vor efectua pe tabele (*vedeTabela(...)*, *modificaTabela(...)*, *stergeTabela(...)*, *adaugaAbonat(...)*, *adaugaCompanieTelefonie(...)*, *adaugaAbonament(...)* etc.).

- Fișierul *ProiectController.java* ce controlează și gestionează comportamentul elementelor ferestrei principale descrise în fișierul *Proiect.fxml*. Metodele implementate în controller fac legătura între elementele cu care interacționează utilizatorul (butoane, liste de tip „drop-down” și formulare) și operațiile implementate în fișierul *DBOperations.java*. De asemenea, tot în *ProiectController.java* au loc și inițializările unor noi ferestre (descrise prin intermediul fișierelor *AdaugaAbonament.fxml*, *AdaugaAbonat.fxml*, *AdaugaCompanieTelefonie.fxml*, *ModificaAbonament.fxml*, *ModificaAbonat.fxml*, *ModificaCompanieTelefonie.fxml*) care să permită utilizatorului introducerea și modificarea datelor în baza de date.

- Fișierele *Abonamente.java*, *Abonat.java*, *CompanieTelefonie.java* care implementează constructorii claselor aferente și metodele de tip getter și setter. Aceste clase urmează tiparul de proprietăți JavaFX, care sunt frecvent utilizate în aplicații pentru a facilita legarea datelor și pentru a permite actualizarea automată ale elementelor de interfață grafică atunci când datele subiacente se modifică.



### 4.3.2. Implementarea funcțiilor

#### Cum se realizează conexiunea?

Prin instrucțiunea *jb.connect()* de la începutul fiecărei metode, ce are ca scop realizarea de operații CRUD asupra bazei de date, a fișierul *ProiectController.java* ne legăm la funcția *connect()* din clasa *DBOperations*:

```
20 public void connect() throws ClassNotFoundException, SQLException, Exception {
21     try {
22         Class.forName("com.mysql.cj.jdbc.Driver");
23         con = DriverManager.getConnection("jdbc:mysql://localhost:3306/abonamentdb?useSSL=false", "root", " ");
24     } catch (ClassNotFoundException cnfe) {
25         error = "ClassNotFoundException: Nu s-a gasit driverul bazei de date.";
26         throw new ClassNotFoundException(error);
27     } catch (SQLException sqle) {
28         error = "SQLException: Nu se poate conecta la baza de date.";
29         throw new SQLException(error);
30     } catch (Exception e) {
31         error = "Exception: A aparut o exceptie neprevazuta in timp ce se stabilea legatura la baza de date.";
32         throw new Exception(error);
33     }
34 } // connect()
```

Obiectul “con” este cel prin intermediul căruia se realizează toate operațiile pe baza de date. În funcția *connect()*: se încarcă driver-ul de MySQL. *DriverManager* conține mai multe drivere de conectică, mai multe SGBD-uri.

Prin funcția *getConnection(...)* apelăm funcția de conectare ce are ca atribute: baza de date realizată în MySQL, user-ul și parola utilizatorului.

Totodată, există și o funcție *disconnect()* ce realizează *close()* pe obiectul de conexiune *con*:

```
68 public void disconnect() throws SQLException {
69     try {
70         if (con != null) {
71             con.close();
72         }
73     } catch (SQLException sqle) {
74         error = ("SQLException: Nu se poate inchide conexiunea la baza de date.");
75         throw new SQLException(error);
76     }
77 } // disconnect()
```

## Cum se afișează datele?

Codul pentru preluarea datelor din tabela dorită, regăsit în fișierul *DBoperations.java* este următorul:

```
34 public ResultSet vedeTabel(String tabel) throws SQLException, Exception {
35     ResultSet rs = null;
36     try {
37         String queryString = ("select * from `abonamentedb`.`" + tabel + "`");
38         Statement stmt = con.createStatement();
39         rs = stmt.executeQuery(queryString);
40     } catch (SQLException sqle) {
41         error = "SQLException: Interogarea nu a fost posibila.";
42         throw new SQLException(error);
43     } catch (Exception e) {
44         error = "A aparut o exceptie in timp ce se extrageau datele.";
45         throw new Exception(error);
46     }
47     return rs;
48 }
```

Se creează instrucțiunea de interogare SQL: *select \* from `abonamentedb`*.

Se realizează un statement pe conexiunea la baza de date, apoi se executa interogarea efectiva (linia 39).

Codul aferent legăturii dintre comportamentul unui buton de încărcare a datelor și preluării acestora din baza de date, cu ajutorul metodei menționate mai sus, aflat în fișierul *ProiectController.java*, este umătorul:

```
201 private void incarcaAbonati(ActionEvent event) throws SQLException, Exception {
202     jb.connect();
203     dateAbonati = FXCollections.observableArrayList();
204     ResultSet rs = jb.vedeTabel("abonati");
205     while (rs.next()) {
206         dateAbonati.add(new Abonat(rs.getInt("idAbonat"), rs.getString("nume"), rs.getString("prenume"), rs.getString("cnp"), rs.getString("adresa")));
207     }
208     atribut_idAbonat.setCellValueFactory(new PropertyValueFactory<>("idAbonat"));
209     atribut_NumeA.setCellValueFactory(new PropertyValueFactory<>("nume"));
210     atribut_PrenumeA.setCellValueFactory(new PropertyValueFactory<>("prenume"));
211     atribut_CNPA.setCellValueFactory(new PropertyValueFactory<>("cnp"));
212     atribut_AdresaA.setCellValueFactory(new PropertyValueFactory<>("adresa"));
213     tabela_Abonati.setItems(null);
214     tabela_Abonati.setItems(dateAbonati);
215     jb.disconnect();
216 }
217 @FXML
218 private void startAduagaAbonat() throws IOException {
219     Stage stage = new Stage();
220     AnchorPane root = (AnchorPane)FXMLLoader.load(getClass().getResource("AduagaAbonat.fxml"));
221     Scene scene = new Scene(root);
222     stage.setScene(scene);
223     stage.show();
224 }
```

Fereastra principală a aplicației, în care sunt afișate datele tabelor, este caracterizată prin intermediul fișierului *Proiect.fxml* care leagă toate butoanele și attributele reprezentate în mod grafic, de metodele aferente implementate în *ProiectController.java*.



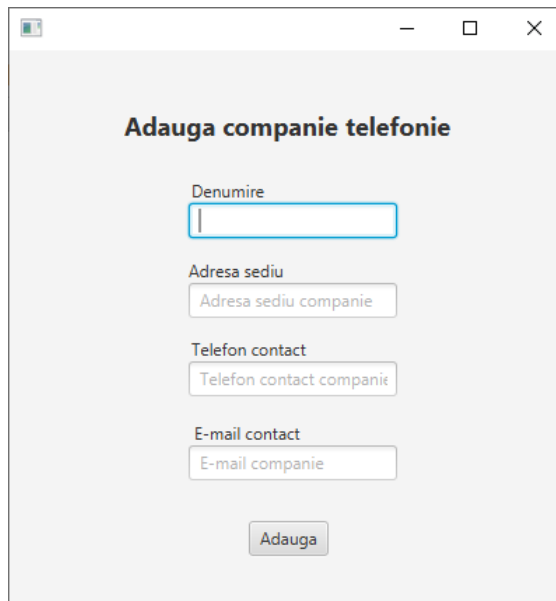
## Cum se realizează adăugarea datelor?

Odată apăsând butonul “*Adaugă abonat*”, “*Adaugă companie*” sau “*Adaugă abonament*” după caz, o nouă fereastră va fi deschisă pentru ca utilizatorul să completeze în câmpurile corespunzătoare, datele noii intrări.

Ferestrele de adăugare sunt descrise grafic în fișierele *AdaugaAbonat.fxml*, *AdaugaCompanieTelefonie.fxml* respectiv *AdaugaAbonament.fxml* ce sunt legate la metodele corespunzătoare implementate în fișierul *ProiectController.java*.

```
257 private void startAdaugaCompanieTelefonie() throws IOException {
258     Stage stage = new Stage();
259     AnchorPane root = (AnchorPane)FXMLLoader.load(getClass().getResource("AdaugaCompanieTelefonie.fxml"));
260     Scene scene = new Scene(root);
261     stage.setScene(scene);
262     stage.show();
263 }
264
265 @FXML
266 private void adaugaCompanieTelefonie() throws SQLException, Exception {
267     String denumire = inputDenumireC.getText();
268     String adresaSediu = inputAdresaSediuC.getText();
269     String telefonContact = inputTelefonContactC.getText();
270     String emailContact = inputEmailContactC.getText();
271
272     jb.connect();
273     jb.adaugaCompanieTelefonie(denumire, adresaSediu, telefonContact, emailContact);
274     jb.disconnect();
275 }
```

Observăm că este apelată metoda *adaugaCompanieTelefonie(..)* din *DBOperations.java*, unde asemenea operațiilor anterioare, se verifică dacă avem conexiune cu baza de date și se execută o comandă specifică MySQL (*insert into companie\_telefonie(...)*).



The image shows a JavaFX window titled "Adauga companie telefonie". Inside the window, there are four text input fields arranged vertically. The first field is labeled "Denumire" and is empty. The second field is labeled "Adresa sediu" and contains the placeholder text "Adresa sediu companie". The third field is labeled "Telefon contact" and contains the placeholder text "Telefon contact companie". The fourth field is labeled "E-mail contact" and contains the placeholder text "E-mail companie". Below these fields is a button labeled "Adauga".

## Cum se realizează modificarea datelor?

În cazul modificării unei intrări, va trebui selectat în tabelul rândul corespunzător intrării din baza de date ce se dorește a fi modificat, apoi apăsat butonul “Modifica abonat”, “Modifica abonament” sau “Modifica companie”, după caz. O nouă fereastră se va deschide, ce are la bază un fișier separat .fxml dedicat interfeței grafice (*ModificaAbonat.fxml*, *ModificaAbonament.fxml*, respectiv *ModificaCompanieTelefonie.fxml*). Asemenea adăugării, toate aceste fișiere .fxml sunt legate la metodele corespunzătoare din controller.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.control.Button?>
4 <?import javafx.scene.control.ComboBox?>
5 <?import javafx.scene.control.Label?>
6 <?import javafx.scene.layout.AnchorPane?>
7 <?import javafx.scene.text.Font?>
8
9 Bind to grammar/schema...
10 <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="507.0" prefWidth="398.0" xmlns="http://javafx.com/fxml"
11 <children>
12 <Label layoutX="91.0" layoutY="82.0" text="Modifica abonament">
13 <font>
14 <font name="System Bold" size="22.0" />
15 </font>
16 </Label>
17 <Button layoutX="168.0" layoutY="313.0" mnemonicParsing="false" onAction="#modificaAbonament" text="Modifica" />
18 <Label layoutX="125.0" layoutY="170.0" text="Abonat" />
19 <Label layoutX="125.0" layoutY="210.0" text="Companie Telefonie" />
20 <ComboBox fx:id="combobox_abonatim" layoutX="124.0" layoutY="187.0" prefWidth="150.0" />
21 <ComboBox fx:id="combobox_companiitelefoniem" layoutX="124.0" layoutY="236.0" prefWidth="150.0" />
22 </children>
23 </AnchorPane>
```

Spre exemplu în cazul tabelii abonamente, butonul *Modifica* are ca acțiune apelarea metodei *modificaAbonament(...)* din *ProiectController.java*:

```
421 private void startModificaAbonament() throws IOException {
422     Abonamente abonamente = tabela_Abonamente.getSelectionModel().getSelectedItem();
423
424     if(abonamente!=null)
425     {
426         x=abonamente.getIdAbonament();
427         Stage stage = new Stage();
428         AnchorPane root = (AnchorPane)FXMLLoader.load(getClass().getResource("ModificaAbonament.fxml"));
429         Scene scene = new Scene(root);
430         stage.setScene(scene);
431         stage.show();
432     }else {
433         System.out.println("Nu s-a putut face modificarea abonamentului!");
434     }
435 }
436 @FXML
437 private void modificaAbonament(ActionEvent event) throws SQLException, Exception {
438     try {
439
440         jb.connect();
441         String[] valori = {String.valueOf(combobox_abonatim.getValue()),String.valueOf(combobox_companiitelefoniem.getValue())};
442         String[] campuri = {"idAbonat", "idCompanieTelefonie"};
443         jb.modificaTabela("abonamente", "idAbonament",x ,campuri, valori);
444         jb.disconnect();
445     } catch (Exception e) {
446         System.out.println("Error!");
447         return;
448     }
449     jb.disconnect();
450 }
451 }
```

Prin intermediul metodei *modificaAbonament(...)* se apelează metoda *modificaTabela(...)* ce creează și execută statement-ul corespunzător instrucțiunii de *update*.

```
138 public void modificaTabela(String tabela, String primaryKey, long ID, String[] campuri, String[] valori)
139     throws SQLException, Exception {
140     String update = "update " + tabela + " set ";
141     String temp = "";
142     if (con != null) {
143         try {
144             for (int i = 0; i < campuri.length; i++) {
145                 if (i != (campuri.length - 1)) {
146                     temp = temp + campuri[i] + "=" + valori[i] + ", ";
147                 } else {
148                     temp = temp + campuri[i] + "=" + valori[i] + " where " + primaryKey + " = '" + ID + "'";
149                 }
150             }
151             update = update + temp;
152             // create a prepared SQL statement
153             Statement stmt;
154             stmt = con.createStatement();
155             stmt.executeUpdate(update);
156         } catch (SQLException sqle) {
157             error = "ExceptieSQL: Reactualizare nereusita; este posibil sa existe duplicate.";
158             throw new SQLException(error);
159         }
160     } else {
161         error = "Exceptie: Conexiunea cu baza de date a fost pierduta.";
162         throw new Exception(error);
163     }
164 } // end of modificaTabela()
```

## 5) Concluzii

Proiectul a implicat crearea a două tabele principale, "*abonati*" și "*companie\_telefonie*", iar relația M:N dintre acestea a fost gestionată cu ajutorul tabelii intermediare "*abonamente*". Acest model a permis o structură flexibilă pentru gestionarea datelor despre abonați și companiile de telefonie asociate.

Interfața oferă utilizatorilor o experiență intuitivă pentru a adăuga, actualiza și șterge informații în baza de date. Funcționalitățile implementate au inclus gestionarea abonamentelor, afișarea detaliilor despre abonați și companii de telefonie, și facilitarea operațiilor de interogare.

## 6) Bibliografie

[1] <https://www.nav.ro/blog/ce-este-mysql/>

[2] <https://www.geeksforgeeks.org/javafx-tutorial/>

[3] <https://docs.oracle.com/javafx/2/swing/overview.htm>

[4] Cursurile PIBD