**Enterprise Invoice Application**

**Coordonator științific**: **Bogdan POSA**

**Absolventă** :

**POPA I. Andreea-Valentina**

**București,**

**2020**

**CONȚINUT PROIECT FINAL**

**CAPITOLUL I –** Introducere....................................................................................pag. 3

**CAPITOLUL II. –** Contribuții personale: .................................................................pag. 5

1. Prezentarea aplicației Enterprise Invoice Application...................pag. 5
2. Implementarea aplicației Enterprise Invoice Application..............pag. 14

**CAPITOLUL III –** Concluzii.....................................................................................pag. 47

**CAPITOLUL I – Introducere:**

*“Programatorul este un creator de universuri pentru care doar el este responsabil. Universuri de complexitate practic nelimitată pot fi create sub forma programelor de calculator.” -* ***Joseph Weizenbaum***

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către **James Gosling** la Sun Microsystems (acum filială **Oracle**) la începutul anilor ʼ90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile, spre exemplu telefon, agenda electronică, etc.

În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intranet-urilor.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos.

Un program ***Java*** compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java (engleză **Java Virtual Machine**, prescurtat **JVM**). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (engleză byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF. În 2006, Sun a anunțat că face disponibilă varianta sa de JVM ca open-source.

Lucrarea finală din cadrul **LinkAcademy**, are ca scop utilizarea cât mai ușoară a programării în ***Java*** și practicarea cât mai rapidă a aceasteia într-un program informatic ce este realizat în programul ***SceneBuilder***, respectiv în limbajul ***JavaFx***.

Pe parcursul lucrării, plecând de la noțiunile generale, teoretice de ***Java Core, MYSQL și JavaFX***, se poate ajunge la un program informatic , o aplicație desktop ușor utilizat de către oricine, nu doar de către programatori, poate fi utilizat și de către contabili, administratori de companie ș.a., realizat în **SceneBuilder,** respectiv **Intellij IDEA**.

În capitolul al doilea, practic exemplificăm aceste noțiuni teoretice prin realizarea unei aplicații desktop numit – **Enterprise Invoice Application**, unde vom detalia etapele de realzare a acestui program.

Iar în ultimul capitol, este vorba despre anumite concluzii legate de această lucrare finală, ce recomandări pot exista, și cât de bine poate fi utilizat și îmbunătățit acest program informatic, viitoarele update-uri mai exact.

**CAPITOLUL II – Contribuții personale :**

În acest capitol se discută despre două subcapitole esențiale lucrării finale și anume faptul că:

1. **Prezentarea aplicației**, unde se va prezenta structura aplicației propriu-zise, și structura bazei de date pe baza noțiunilor învățate până la momentul actual ;
2. **Implementarea aplicației**, în editorul **Intelij IDEA**, dar și în aplicația **Scene Builder**.

Aici se va exemplifica modul cum au fost concepute cele patru formulare în Scene Builder, cât și partea lor de funcționare, mai exact Controarele, partea de stilizare a acestora prin **JavaFX – CSS**, dar și conexiunea la baza de date prin **MySQL Workbench**.

1. **Prezentarea aplicației**:
2. Structura aplicației propriu-zise:

* În aplicația **Enterprise Invoice Application**, au fost create trei pachete (package) :

1. images ;
2. sample ;
3. utils.

În pachetul “images”, apar doar pozele aplicației utilizate mai ales în aplicația de interfață Scene Builder, dar și în partea de **CSS ( Cascading Style Sheets )** .

În pachetul “sample”, apar atât formularele (Delete, ERP, sample și SignUP) toate având extensia **.FXML** .

**FXML** este un limbaj de marcare a interfeței de utilizator bazat pe XML creat de Oracle Corporation pentru definirea interfeței de utilizator a unei aplicații JavaFX.

Oferă o alternativă convenabilă la construirea unor astfel de grafice în cod procedural și este ideal pentru a defini interfața de utilizator a unei aplicații JavaFX, deoarece structura ierarhică a unui document XML este în paralel cu orice este creat sau implementat în FXML poate fi exprimat folosind JavaFX direct.

Pe lângă partea de FXML, aceste formulare ca să fie funcționabile, au nevoie de anumite **Controllere**.

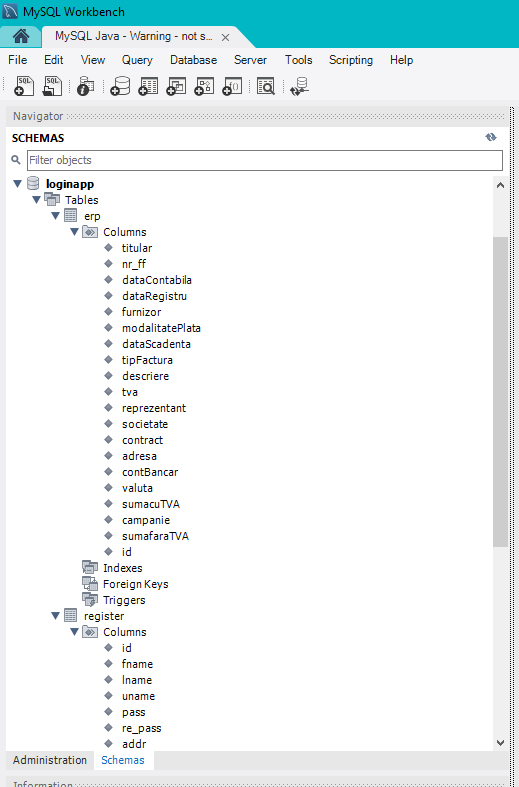
Aceste Controllere implementează clasa **Initializable.** Această clasă este întotdeauna apelat la inițializarea unui controller după ce elementul său rădăcină a fost procesat complet. Această interfață ( Initializable ) a fost înlocuită prin injectarea automată a proprietăților locației și a restuselor în controller. Este recomandat ca atunci când este utilizată clasa Initializable, să fie utilizată ca **adnotare @FXML**.

**Adnotarea @FXML** este cea care etichetează o clasă sau un membru ca fiind accesibilă pentru marcare.

Pe lângă partea de controllere și partea de FXML ( formulare ), avem și partea de stilizare a acestor componente de interfață cu utilizatorul și anume fișierul **„stylesheet.css”.**

Ca și diferență față de **HTML ( HyperText Markup Language)**, în JavaFX acest fișier CSS începe orice schimbare de interfață cu **„ –fx „** .

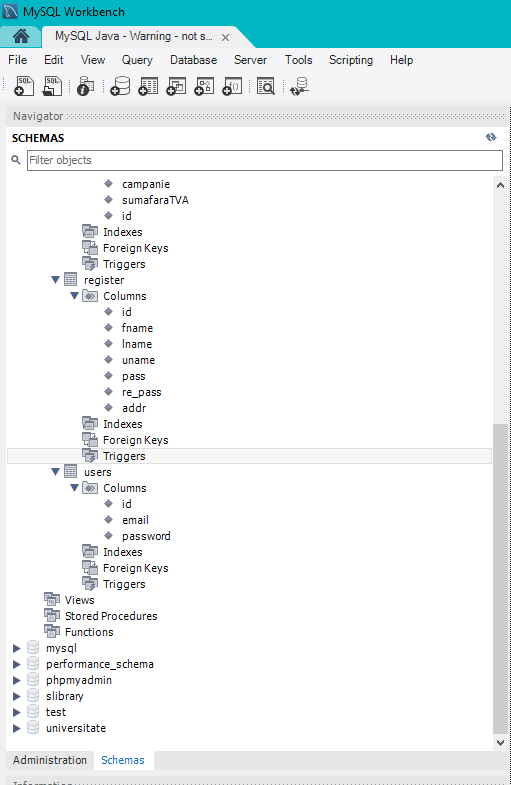
Și, în final, în ultimul pachet numit “utils” . avem efectiv conectarea la baza de date a programului informatic Enterprise Invoice Application.



1. Structura bazei de date :

Proiectul Enterprise Invoice Application are ca și structură a bazei de date următoarele tabele:

Primele două tabele sunt prezentate în stânga, iar ultimul tabel este pe următoarea pagina.



1. Tabela “erp” ;
2. Tabela “register” ;
3. Tabela “users”.

Dacă ar fi să ne luăm după definiția ”bazelor de date relaționale” și anume faptul că informațiile sunt organizate în **tabele**, fiecare **linie** descriind câte un subiect / **entitate**. O linie poartă numele de **înregistrare**. Descrierea fiecărei însușiri a subiectului o constituie un **câmp**.

**Atributele ( Coloanele )** descriu entitățile ( rândurile ) cărora le aparțin.

Atributele se mai folosesc și pentru a deosebi mai multe entități de același tip.

Lipsa unui atribut adecvat se soluționează prin introducerea unui atribut al cărui scop social este identificarea entității (ID).

Ținându-se cont de aceste noțiuni teoretice***, particularizând pe programul informatic Enterprise Invoice Application*** :

**Tabelele** aplicației sunt :

1. erp – ce reprezintă formularul de tip ERP, locul în care o factură de diverse servicii precum chirii, utilități ș.a. este înregistrată și salvată în baza de date ;
2. register – ce reprezintă formularul de tip SignUp (register) unde un utilizator se înregistrează pentru a avea acces la aplicația informatică Enterprise Invoice Application ;
3. users – reprezintă tabela de teste pentru developeri pentru a testa, analiza programul .
4. ***Tabelul de tip ”erp”*** are ca și atribute următoarele :

* ***atributul titular*** :

Reprezintă numele celui care prestează serviciile la o anumită societate. Acesta în baza de date este de tip VARCHAR de mărime maximă 255. not null (NN). În cadrul controller-ului de tip ERP este de tip String ;

* ***atributul nr\_ff*** :

Reprezintă numărul facturii fiscale introduse în programul informatic, pentru a putea ține evidența mai ușor. Acesta în baza de date este de tip INT de mărime maximă 255, NN. În cadrul controller-ului de tip ERP este tot de tipul int ;

* ***atributul dataContabila*** :

Reprezintă data în care este înregistrată factura fiscală la momentul actual. Acesta în baza de date este de tipul DATE. În partea de controller al formularului de tip ERP este de tip LocalDate (cel care modelează data) ;

* ***atributul dataRegistru*** :

Reprezintă data în care factura fiscală a fost emisă. Acesta în vaza de date este de tipul DATE. În partea de controller al formularului de tip ERP este tot de tipul LocalDate ( cel care modelează data) ;

* ***atributul furnizor*** :

Reprezintă numele furnizorului, cel care furnizează diverse servicii către o companie. Acesta în baza de date este de tipul VARCHAR de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul modalitatePlata***:

Reprezintă modalitatea de plată aleasă de către furnizor, de obicei este de tip ”card bancar”. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul dataScadenta*** :

Reprezintă data de expirare a facurii fiscale. Acesta în baza de date este de tipul DATE. În partea de controller al formularului de tip ERP este de tipul LocalDate ( cel care modelează date) ;

* ***atributul tipFactura*** :

Reprezintă tipul facturii fiscale, ca de exemplu: facturi de utilități, facturi de chirii ș.a. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul descriere***:

Reprezintă descrierea facturii fiscale. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul tva*** :

Reprezintă TVA-ul facturii fiscale, este și un câmp calculat. Am ales ca pentru acest program informatic , TVA-ul să aibă doar cota de 19%, restul cotelor fiind adăugate pe viitor. Acesta în baza de date este de tipul INT de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul int ;

* ***atributul reprezentant*** :

Aici se va trece numele celui care are ca responsabilitate factura respectivă. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul societate*** :

Reprezintă numele companiei respective. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul contract*** :

De obicei o factură fiscală nu poate fi emisă fără a exista un contract între cele două părți terțe, mai ales pentru companii. În baza de date este de tipul INT de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul int ;

* ***atributul adresa*** :

Reprezintă adresa furnizorului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul contBancar*** :

Reprezintă contul bancar al furnizorului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul valuta*** :

De obicei facturile înregistrate sunt numai în valuta RON. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul sumacuTVA*** :

Acesta este un câmp calculat, deoarece este format din: sumafaraTVA și TVA-ul facturii ( sumafaraTVA este preluată din factura fiscală fizică). Acesta în baza de date este de tipul INT, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul int ;

* ***atributul campanie*** :

Reprezintă luna și anul de când este creată factura fiscală. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul String ;

* ***atributul sumafaraTVA*** :

Acesta este un câmp unde se va introduce suma fără TVA ( este specificată pe factura fiscală suma). Acesta în baza de date este de tipul INT, de mărime maximă 255. În partea de controller al formularului de tip ERP este de tipul int ;

* ***atributul id*** :

Reprezintă cheia primară a tabelului ”erp”, este not null și auto-increment ( NN ; AI ), ca mărime maximă este 255 , tipul fiind INT. În partea de controller, fiind AutoIncrement (AI), acesta nu se mai implementează.

1. ***Tabelul de tip ”register”*** are ca și atribute următoarele :

* ***atributul id*** :

Reprezintă cheia primară a tabelului ”register” este not null și auto-increment ( NN ; AI ), ca mărime maximă este 255 , tipul fiind INT. În partea de controller, fiind AutoIncrement (AI), acesta nu se mai implementează.

* ***atributul fname*** :

Reprezintă numele de familie al utilizatorului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

* ***atributul lname*** :

Reprezintă prenumele utilizatorului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

* ***atributul uname*** :

Reprezintă numele de utilizator al companiei. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

* ***atributul pass*** :

Reprezintă parola contului creat. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

* ***atribtuul re\_pass*** :

Reprezintă reintroducerea parolei contului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

* ***atributul addr*** :

Reprezintă adresa utilizatorului. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip SignUp este de tipul String ;

1. ***Tabelul de tip ”users”*** are ca și atribute următoarele :

* ***atributul id*** :

Reprezintă cheia primară a tabelului ”users” este not null și auto-increment ( NN ; AI ), ca mărime maximă este 255 , tipul fiind INT. În partea de controller, fiind AutoIncrement (AI), acesta nu se mai implementează.

* ***atributul email*** :

Aici developerul își va introduce contul creat. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip Login este de tipul String ;

* ***atributul password*** :

Aici developerul își va introduce parola aleasă. Acesta în baza de date este de tipul VARCHAR, de mărime maximă 255. În partea de controller al formularului de tip Login este de tipul String .

1. **Implementarea aplicației :**

Pentru a începe acest subcapitol, am hotărât să prezint în două etape și anume:

1. **Partea funcțională** – Prezentarea în detaliu a funcționalității aplicației Enterprise Invoice Application ;
2. **Partea de interfață cu utilizatorul** – Prezentarea vizuală a aplicației în detaliu.
3. **Partea funcțională** :

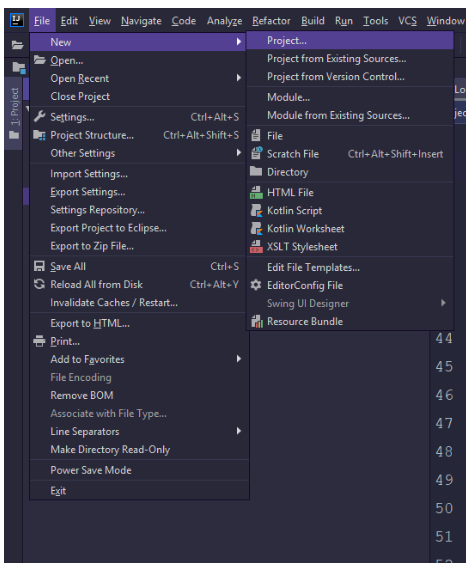
Pentru a implementa în editorul Intelij IDEA, avem de urmat următorii pași esențiali pentru începerea aplicației propriu-zise.

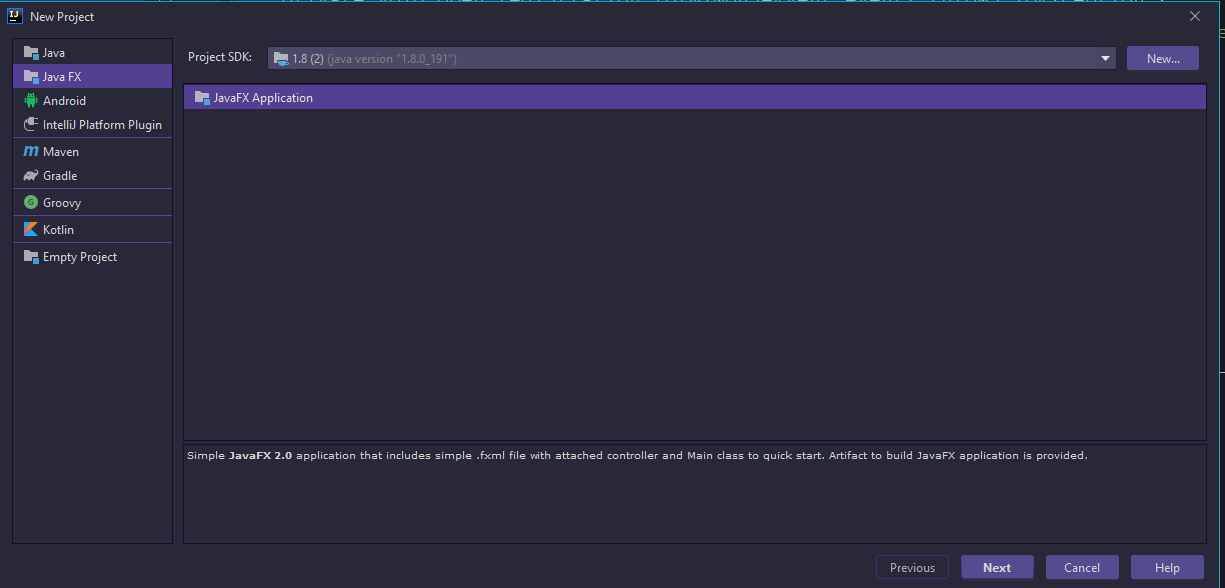
1. Se deschide Intelij IDEA:



1. Din tabul ”File”:

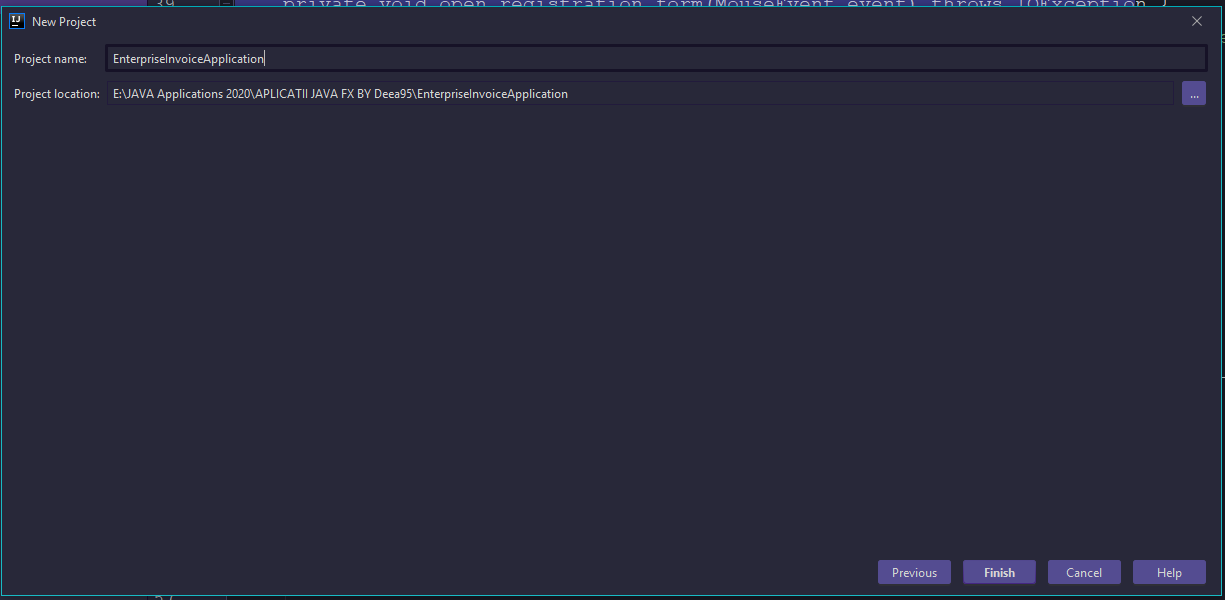
**New -> Project**





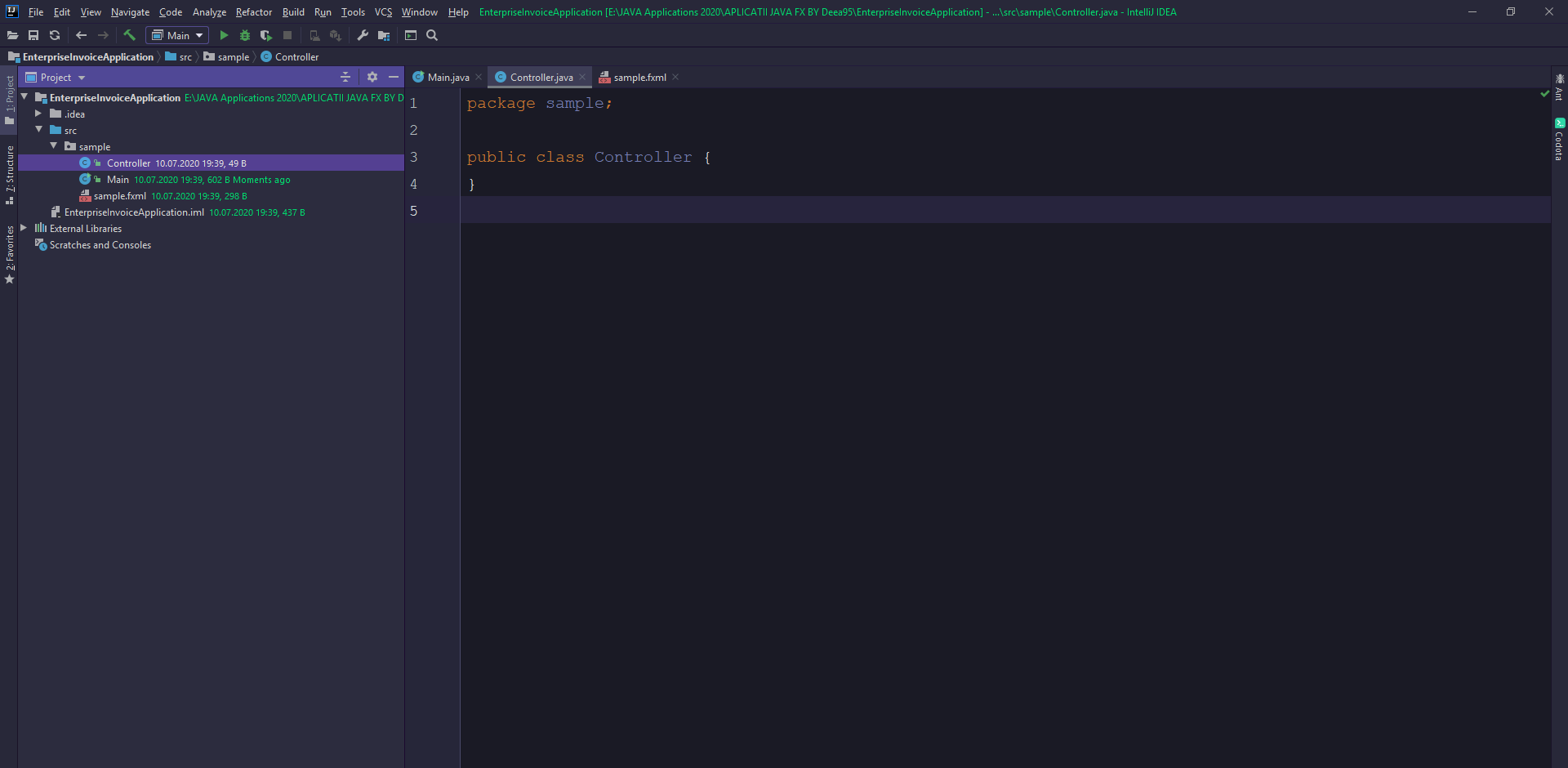
Apoi selectăm **JavaFX -> JavaFX Application** unde **SDK este 1.8** :

Apăsăm pe Next -> Alegem un nume și locația proiectului :



Iar apoi, apăsăm pe **FINISH**.

1. Va apărea cod-ul inițial al aplicației:



1. În clasa **Main.java** , avem următoarele:
2. Modificări inițiale:

* Modificarea titlului aplicației:



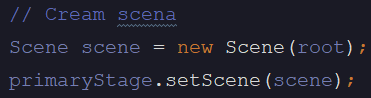
* Cu numele proiectului :



* Modificarea scenei aplicației:



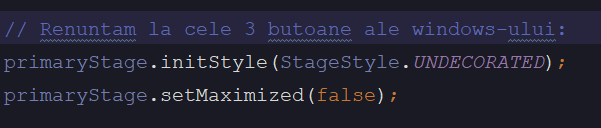
* Cu :



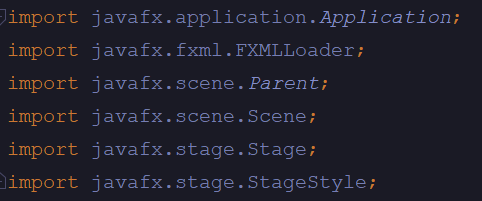
Dimensiunea o alegem în editorul **SceneBuilder**.

1. Adăugările din această clasă:

* renunțarea la cele 3 butoane ale Windows-ului pentru estetică:



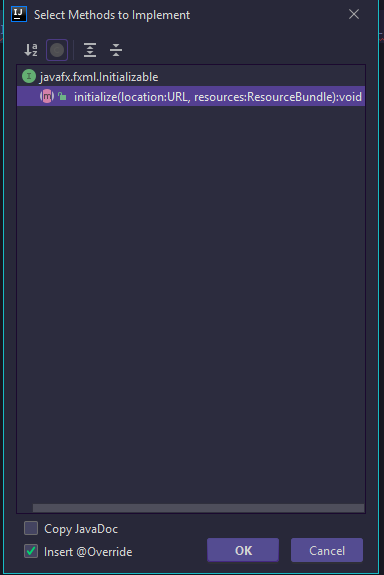
Pentru a funcționa această clasă, se vor face importurile necesare!



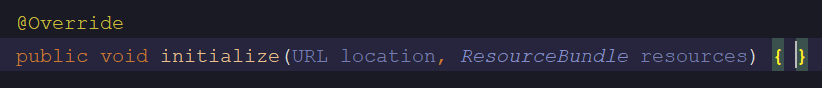
1. În clasa **Main\_Controller.java** avem următoarele :
2. Se va **implementa clasa Initializable** :



După cum se observă, această clasă implementată are o eroare. Pentru a rezolva această eroare, vom apăsa pe **”Implement methods”**.

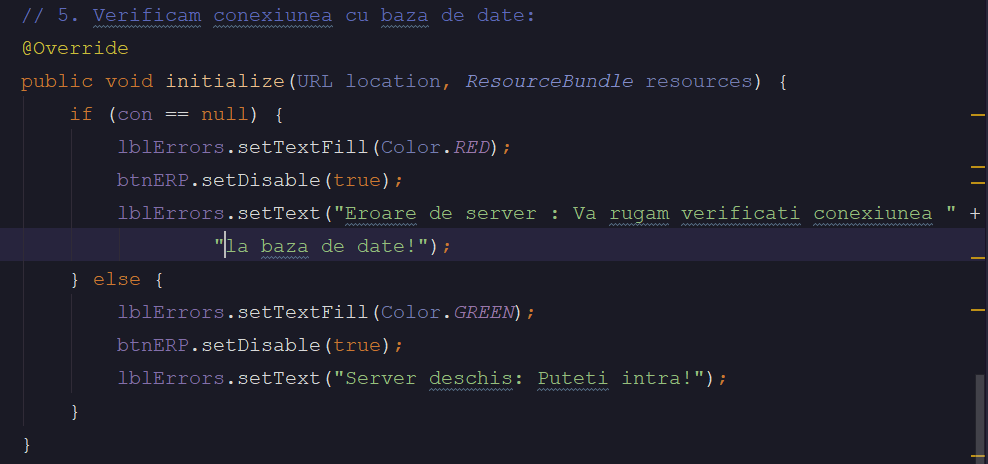


1. Metoda care s-a implementat arată în felul următor:



În această metodă, am creat un **label de eroare** **( lblErrors)**, acesta se va umple cu **roșu** dacă: **”Eroare de server: Vă rugăm verificați conexiunea la baza de date”**, iar cu culoarea **verde** dacă: **”Server deschis: Puteți intra!”**. Am mai pus o micuță **retricție** în legătură cu butonul **btnERP**, acesta nu trebuie să fie activ până când utilizatorul își introduce corect user-ul și parola.

În acest caz, am apelat la două metode pentru **label-ul lblErrors (setTextFill și setText)**, iar pentru butonul **btnERP** am apelat la metoda **setDisable(true)**.



1. Adăugările din această clasă :
2. Se va face import următoarelor pachete:

* Din **pachetul javafx.fxml** :

FXML ;

FXMLLoader ;

Initializable .

* Din **pachetul javafx.scene** :

Parent ;

Scene ;

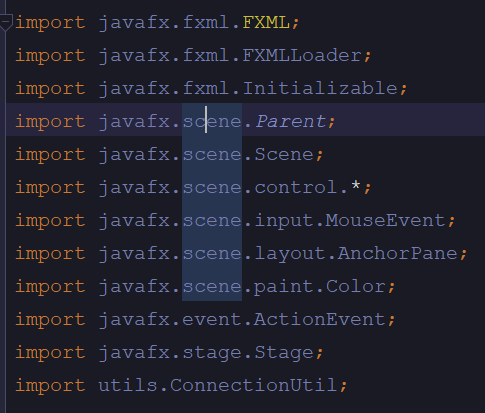
control.\*;

input.MouseEvent ;

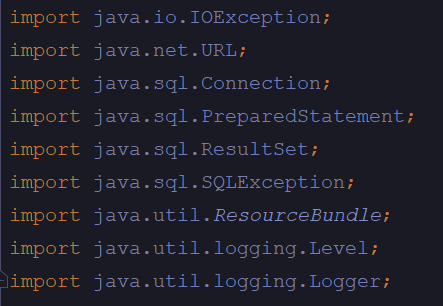
layout.AnchorPane ;

paint.Color .

* ActionEvent din **pachetul javafx.event** :
* Stage din **pachetul javafx.stage** ;
* ConnectionUtil din **pachetul utils**.

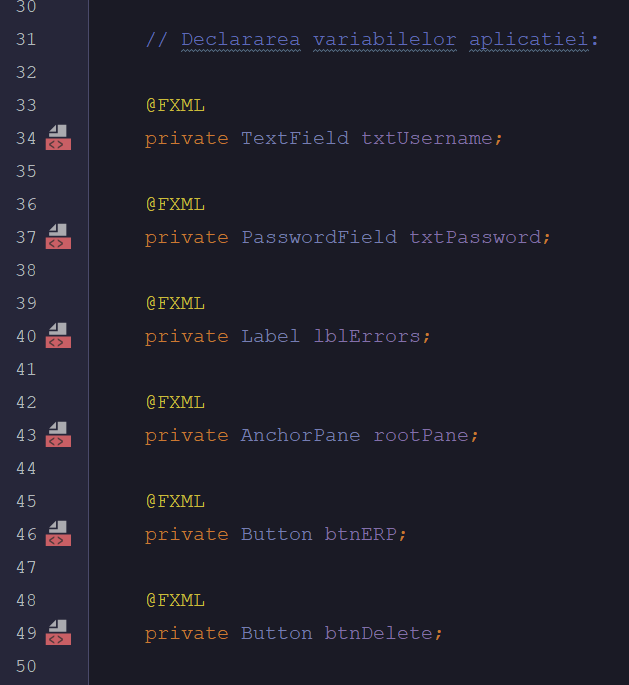


Pe lângă aceste pachete, se va mai importa :



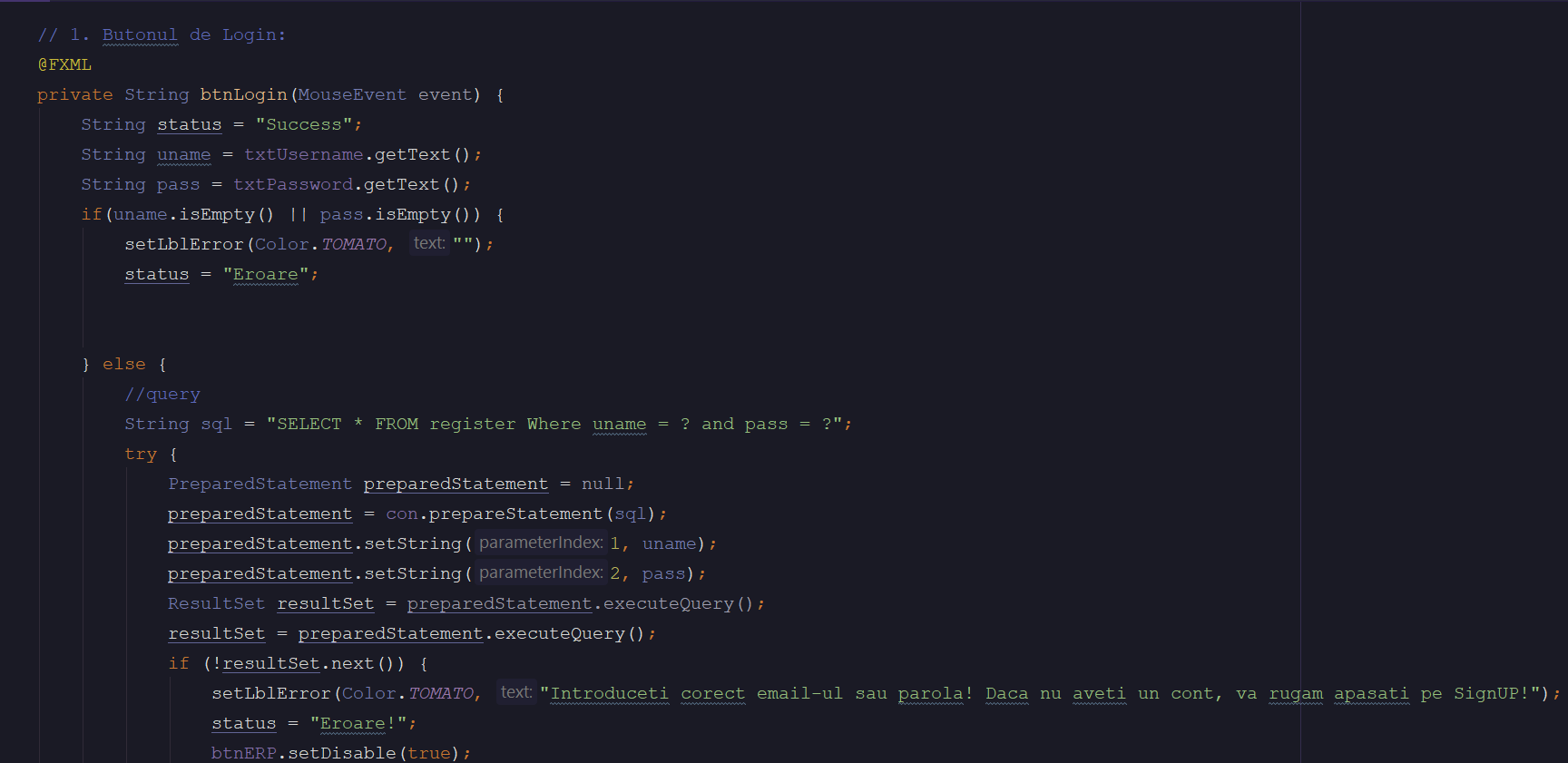
1. După ce s-au făcut toate importurile necesare, se vor declara variabilele clasei **Main\_Controller.java** .

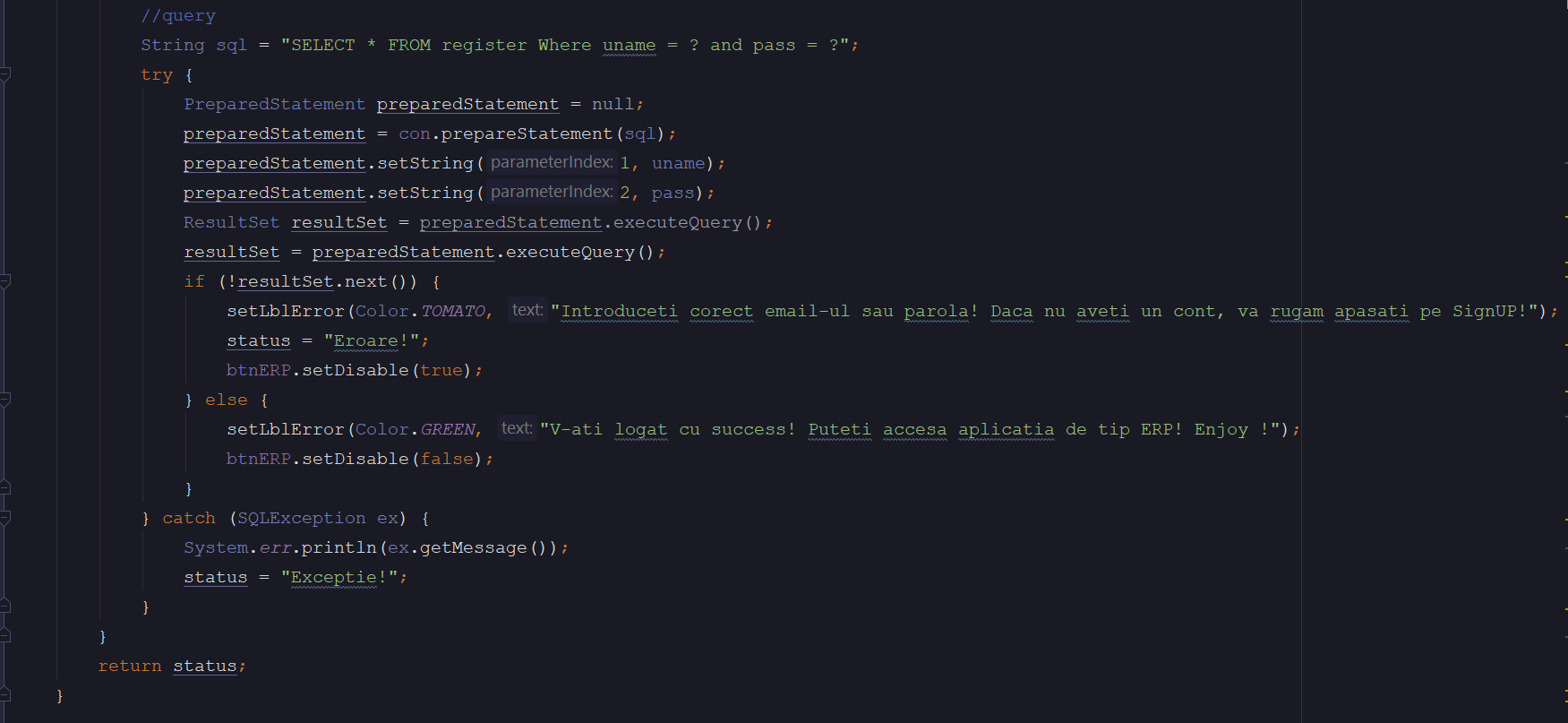
Aceste declarări se pot prelua fie din aplicația de interfață SceneBuilder, fie se pot declara de mână astfel:



Unde:

* **txtUsername** este o variabilă și o casetă text (TextField) unde utilizatorul își introduce acel nickname ales ;
* **txtPassword** este tot o casetă text (TextField) unde utilizatorul își introduce parola aleasă ;
* **lblErrors** este un Label în care vedem dacă este făcută conexiunea la baza de date sau nu ;
* **rootPane** este de tip AnchorPane și este utilizat pentru a crea noua scenă aleasă ;
* **btnERP** este butonul în care în această clasă am efectuat legătura la o nouă scenă și anume formularul ERP.fxml ;
* **btnDelete** este butonul în care în această clasă am efectuat legătura la o nouă scenă și anume formularul Delete.fxml.

1. Funcționalitatea clasei, presupune :
2. Butonul de **LOGIN** :



1. În acest buton, am apelat la evenimentul de tip **MouseEvent** :

**@FXML**

**private String btnLogin (MouseEvent event)**

1. Am declarat cele două variabile ce sunt utilizate în baza de date în tabelul **”users„** și anume :

**String uname = txtUsername.getText();**

**String pass = txtPassword.getText();**

Le-am declarat sub formă de String, deoarece amândouă în baza de date sunt de tipul **TEXT**.

Am apelat la metoda **getText()**, deoarece această metodă ajută la extragerea textului introdus și introducerea textului respectiv în baza de date.

Apoi, am creat o ”restricție” în care dacă cele două variabile declarate uname și pass sunt goale (apelate prin metoda **isEmpty()** ), atunci se va apela o metodă creată de mine și anume: **setLblError** (unde setează culoarea **roșie**, iar textul este gol), unde statusul este ”eroare”.

Metoda **setLblError()** are 2 parametrii: **Color color** și **String text** :

* îi setăm culoare label-ului lblErrors cu color ;
* îi setăm text label-ului lblErrors cu text ;
* afișăm textul ;
* setăm ca butonul btnERP să fie dezactivat .

Revenim la acel IF și anume: dacă cele două variabile declarate uname și pass nu sunt goale, atunci vom apela la introducerea acestora în baza de date.

Query-ul este:

**String sql = ”SELECT \* FROM register WHERE uname = ? and pass = ?” ;**

De ce am apelat la tabela ”register” și nu la ”users” ? Deoarece vrem să fie incluse și conturile noi create, nu doar contul developerului.

Se va implementa un bloc de tip **try....catch ,** unde apelăm la **clasa PreparedStatement** (**execută interogarea SQL** în acest **obiect preparedStatement** și returnează obiectul lui **ResultSet** generat de o interogare). Inițial se declară cu null acel obiect din **clasa PreparedStatement.**

**PreparedStatement preparedStatement = null ;**

**preparedStatement = con.preparedStatement(sql) ;**

Iar acum declarăm cele două variabile uname și pass așa cum au fost declarate și în baza de date. Acestea sunt declarate prin metoda setString() , deoarece în baza de date sunt declarate de tipul TEXT.

Se apelează clasa ResultSet cu obiectul resultSet unde pentru obiectul preparedStatement vom apela metoda executeQuery().

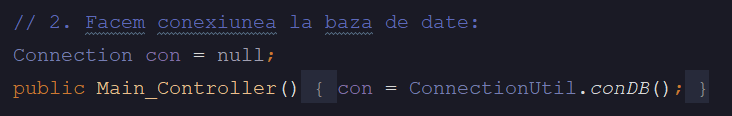
**ResultSet resultSet = preparedStatement.executeQuery();**

Dacă acest resultSet ( eliberează acest ResultSet, baza de date a obiectului și resursele JDBC imediat), apelează metoda .next() și este negată, atunci se va apela metoda setLblError unde i se va seta culoarea roșie, iar ca text: ”Introduceți corect email-ul sau parola! Dacă nu aveți cont, vă rugăm apăsați pe SignUP!”, statusul va fi ”Eroare” și butonul ERP să fie dezactivat. Altfel, se va apela tot metoda setLblError unde de data asta culoarea va fi verde , iar textul : ”V-ați logat cu succes! Puteți accesa aplicația de tip ERP! Enjoy !”, unde de data aceasta butonul ERP va fi vizibil și poate fi accesat.

Toate aceste detalii de mai sus sunt pe partea de try. Pe partea de catch, de obicei se pune excepția.

La final, se va returna statusul.

Legat de acel ”con”, se va face mai jos conexiunea la baza de date :



1. Butonul de **SignUP** :

În acest buton:

* se va apela **evenimentul ActionEvent** ;
* se va crea **obiectul de tip Parent** :

**Parent signupParent;**

* prin acest obiect se va putea importa ***formularul SignUp.fxml*** . Deoarece acest formular este realizat pe un ***AnchorPane***, atunci, se va face cast pentru a nu avea probleme. La fel ca în **Main.java**, se va face un ***Stage***, unde i se va seta un nume și se va face vizibil acest stage :

**Scene nscene = new Scene(signupParent);**

**Stage mainWindow ;**

**mainWindow = (Stage) rootPane.getScene().getWindow();**

**mainWindow.setTitle(”Sign UP”);**

**mainWindow.setScene(nscene);**

**btnERP.setDisable(true);**

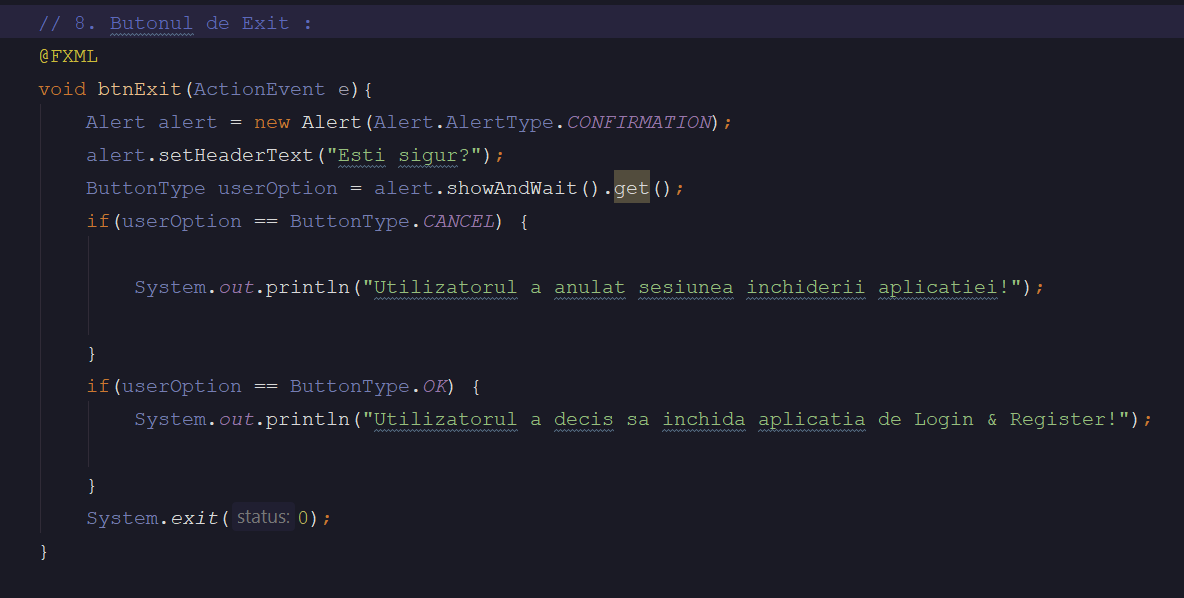
1. Butonul **ERP ( Enterprise Resource Planning)** :

****Ca și butonul de SignUP, în loc să fie importat formularul de SignUP, se va importa formularul de ERP, și se va seta titlul ”ERP System”.

1. ****Butonul de **DELETE**:

Ca și celelalte butoane, exact pe același principiu se va importa formularul de Delete.fxml.

1. Butonul de **EXIT**:

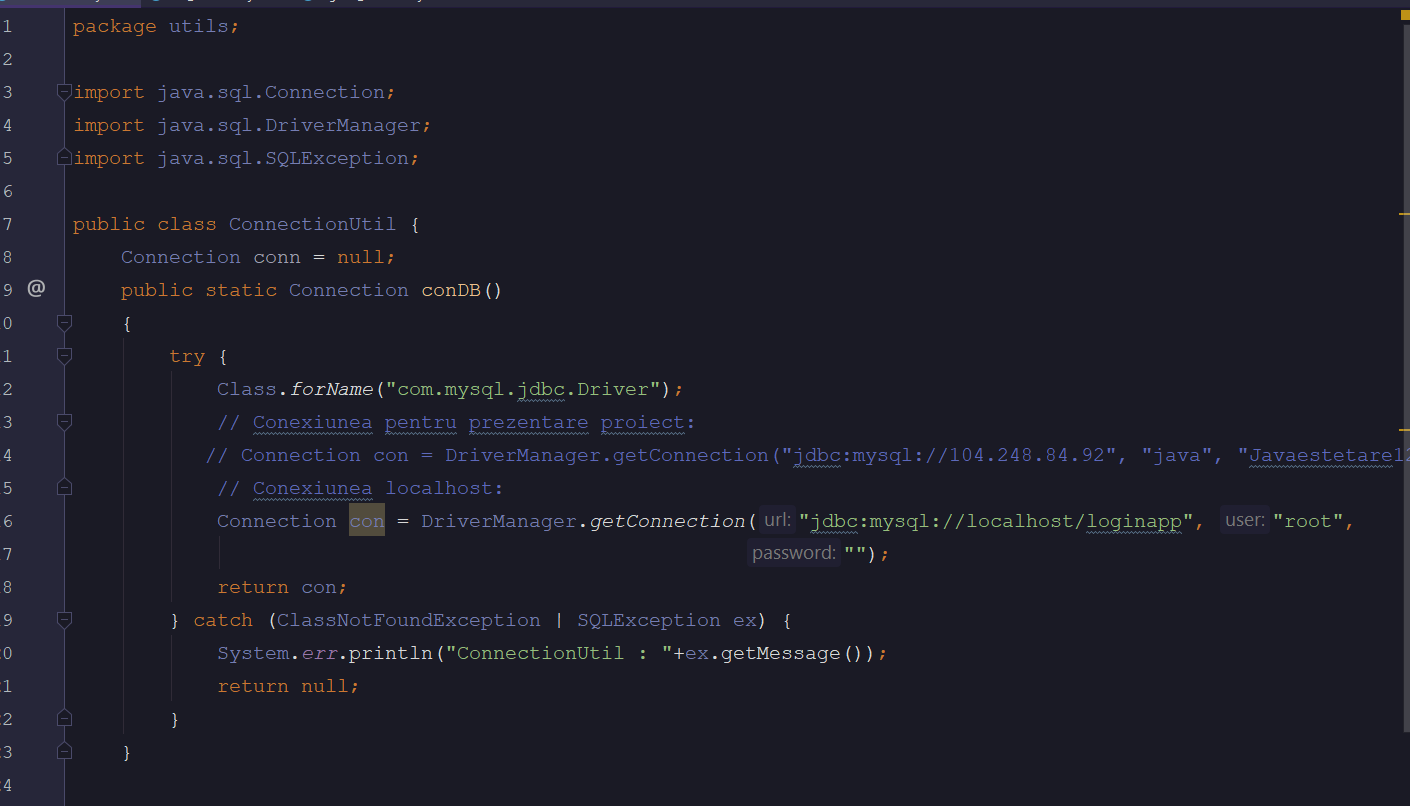
****

Aici am ales să implementez pe bază de Alert ; am ales o alertă de tip **CONFIRMATION**. Dacă utilizatorul alege alerta de tip **CANCEL**, se va afișa un text. Același tip de text se va afișa și dacă apasă pe **OK**. La final, se va pune:

**System.exit(0);**

Aceste două clase **Main.java** și **Main\_Controller.java** sunt clasele de început ale programului.

1. Se va începe cu clasa **ConnectionUtil.java** :

****

După cum se observă în poza de mai sus, se va crea o metodă conDB() în care se face conexiunea la baza de date.

Pentru a face conexiunea la o bază de date în Java, se va utiliza metoda:

**Class.forName(”com.mysql.jdbc.Driver”);**

Aici este specificată locația Driver-ului utilizat de MySQL.

Pentru că, la cursul de la LinkAcademy avem specificată baza de date respectivă, am făcut următoarea sintaxă pentru conexiunea la baza de date :

**Connection con = DriverManager.getConnection(”jdbc:mysql//104.248.84.92”, ”java”, ”Javaestetare123!”);**

Aici specificăm JDBC-ul, IP-ul de la conexiune, în cazul nostru: 104.248.84.32, utilizator: ”java”, parola: ”Javaestetare123!”.

Unde se va returna conexiunea:

**return con;**

Dacă în schimb, este lucrat pe localhost, pentru URL: ”jdbc:mysql://localhost/loginapp”, user : ”root”, paorla : ” ”.

1. Clasa **Detele\_Controller.java** :

* În acest caz, pașii de realizare sunt cam aceeiași ca și în clasa **Main\_Controller.java** :
* se implementează ***clasa Initializable*** ;
* se declară ***variabilele*** utilizate :

**Button btnIesire ;**

**TextField txtUsernameID ;**

**PasswordField txrPasswordD;**

**Button btnBackLoginD;**

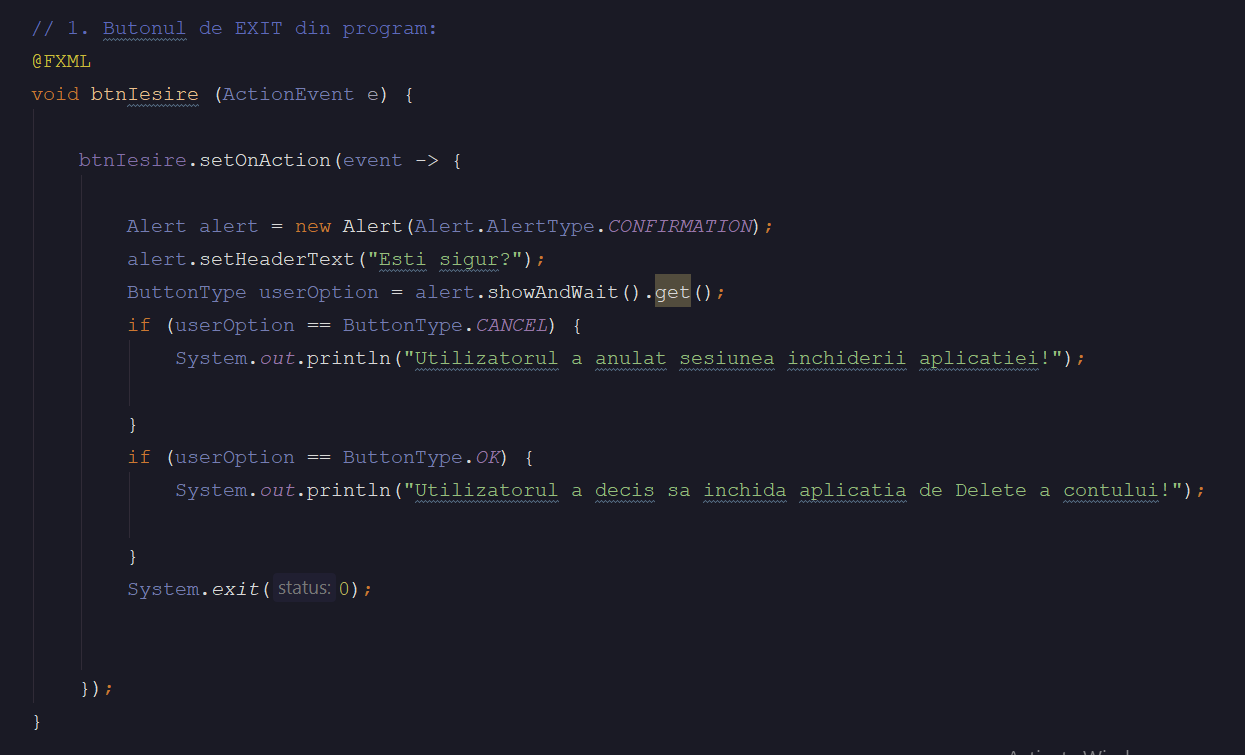
**Button btnCancelD ;**

**Button btnDeleteA ;**

**AnchorPane rootPane.**

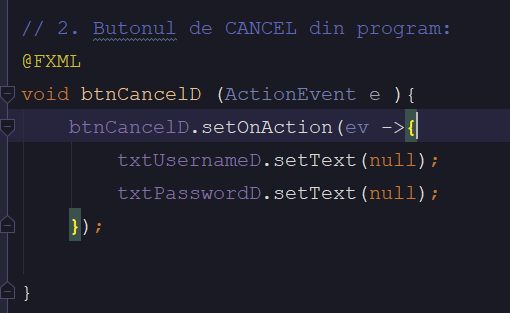
* Funcționalitatea clasei ar fi următoarea :

1. Pentru **butonul de ieșire (btnIesire)**, modul de realizare este exact același ca și în clasa **Main\_Controller** (btnExit în cazul lor) :

****

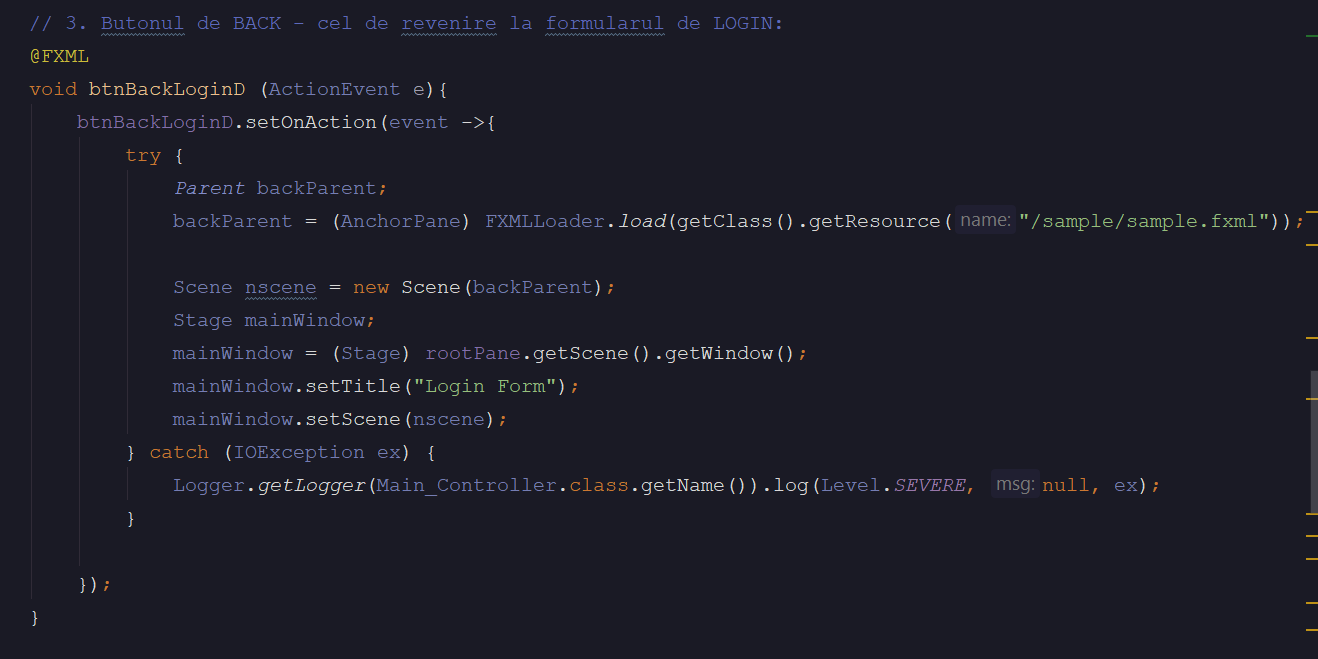
1. Butonul de **CANCEL** din acest formular :

Cu ajutorul metodei: **setText(null)** , setează tuturor variabilelor valoarea ” ” (**null**).



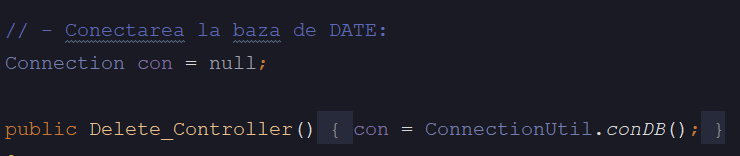
1. Butonul de **BACK** – ***cel de revenire la formularul de LOGIN*** :

Acest buton a fost gândit în felul următor : după ce contul a fost șters definitiv din baza de date, se poate da revenire la ecranul principal, adică LOGIN.

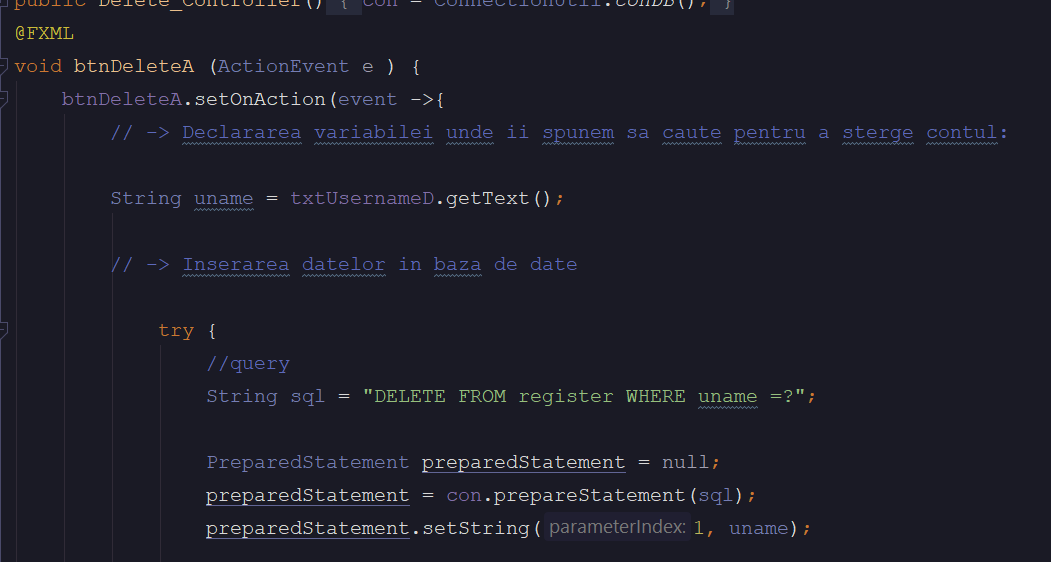


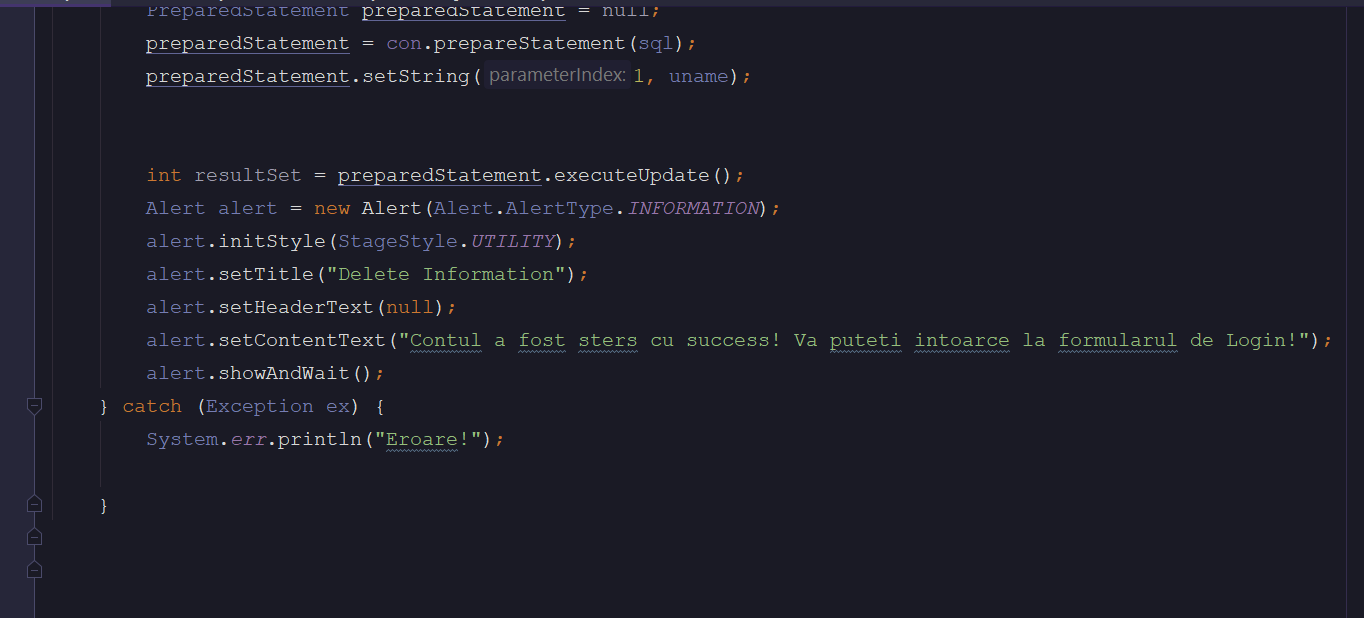
1. Butonul **DELETE** din acest formular :

Pentru început se va face ***conexiunea la baza de date*** :



Pasul următor este apelarea evenimentului de tip ***ActionEvent***. Declarăm variabila necesară ștergerii contului și anume uname :





În blocul de instrucțiune ***TRY...CATCH***, în partea de ***TRY***, la fel ca și în clasa **Main\_Controller.java** , pașii rămân la fel, doar că ***intrucțiunea SQL*** este alta:

**String sql = ”DELETE FROM register WHERE uname = ? ” ;**

Conturile se vor **șterge** pe bază de **username (uname)** , deoarece ID-ul este de tip **AutoIncrement**.

1. Clasa **SignUP\_Controller.java** :

* În acest caz, pașii de realizare sunt cam aceeași ca și în clasele prezentate până acum:
* se implementează clasa Initializable ;
* se declară variabilele utilizate :

**AnchorPane rootPane ;**

**Button btnCancel ;**

**Button btnIesire ;**

**TextField txtFirstName ;**

**TextField txtLastName ;**

**TextField txtUsernameRegister ;**

**TextField txtPasswordRegister ;**

**TextField txtRetypePass ;**

**TextArea taAddress ;**

**Button btnBack ;**

**TextField txtID .**

* Funcționalitatea acestei clase ar fi următoarea :

1. Pentru **butonul de ieșire (btnIesire)**, modul de realizare esre exact același ca și în clasa Main\_Controller (btnExit în cazul lor).

****

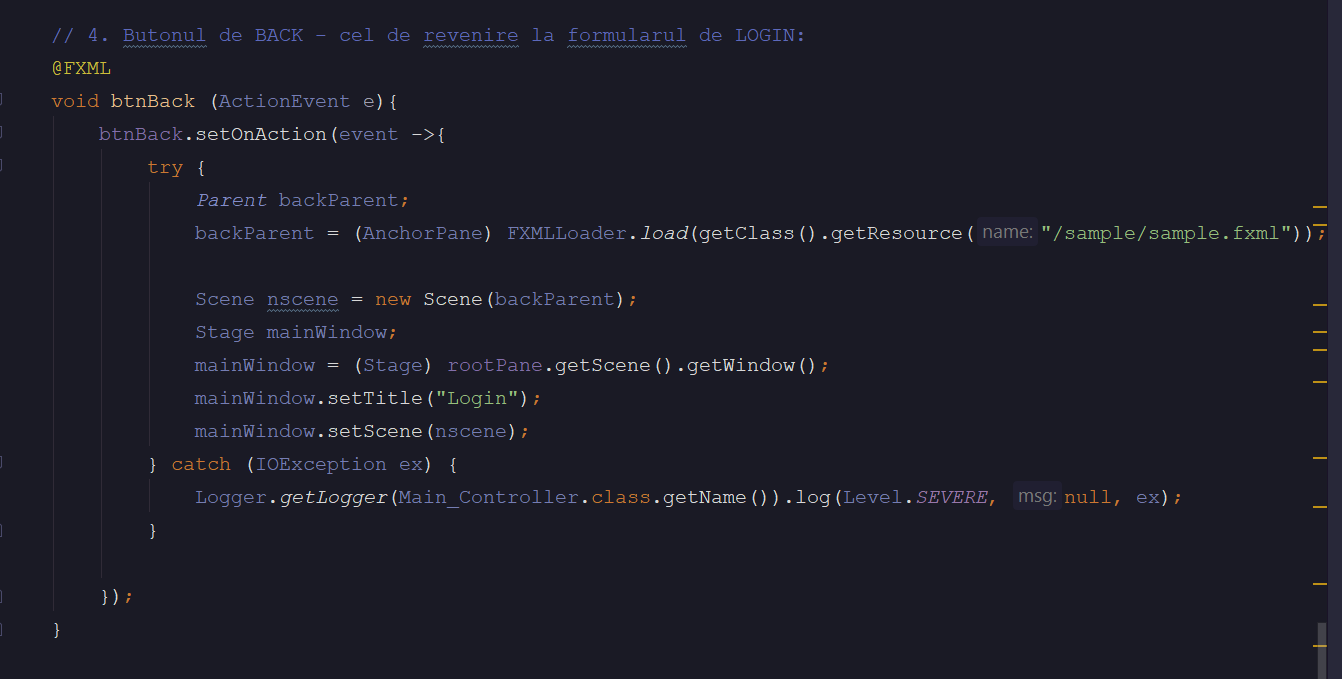
1. Butonul **CANCEL** din acest formular :

Cu ajutorul metodei: **setText(null)**, setează tuturor variabilelor valoarea **null**.



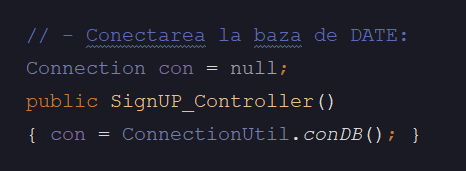
1. Butonul **BACK** – ***cel de revenire la formularul de LOGIN*** :

Acest buton a fost găndit în felul următor : după ce contul a fost creat cu succes în baza de date, se poate da revenire la formularul de LOGIN.

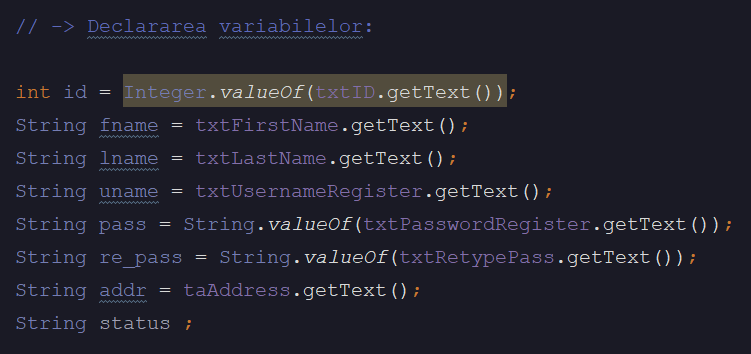


1. Butonul **REGISTER** din acest formular :

Pentru început se face conexiunea la baza de date :



Pasul următor este apelarea evenimentului de tip ***ActionEvent***. Declarăm variabilele necesare exact așa cum au fost declarate și în baza de date:



Dacă variabilele declarate mai sus sunt goale, atunci se va da un ***AlertType*** de tip **ERROR** în care va apărea acest mesaj: ”Vă rugăm completați toate datele !”. Altfel, se va adăuga următoarea instrucțiune SQL :

**String sql = ”INSERT INTO register (id, fname, lname, uname, pass, re\_pass, addr) VALUES (?,?,?,?,?,?,?)” ;**

La fel ca și la celelalte clase prezentate, se va declara cu metoda ***preparedStatement.setString()*** modul în care variabilele au fost declarate în baza de date.

La final se va face un ***AlertType*** dar de data aceasta de tip **INFORMARE**, cu acest mesaj: ”Cont creat cu succes! Vă rugăm reveniți în formularul de Login. ”.

1. Clasa **ERP\_Controller.java** :

* În acest caz, pașii de realizare sunt cam aceeași ca și în clasele prezentate de până acum :
* se implementează clasa Initializable ;
* se declară variabilele utilizate :

**Button btnIesire ;**

**AnchorPane rootPane ;**

**TextField txtTitular ;**

**TextField txtNrFF ;**

**TextField txtSocietate ;**

**TextField txtContract ;**

**DatePicker dpDataContabila ;**

**DatePicker dpDataRegistru ;**

**TextField txtFurnizor ;**

**TextField txtAdresaFurnizor ;**

**TextArea taDescriere ;**

**TextField txtReprezentant ;**

**TextField txtContBancar ;**

**TextField txtModalitatePlata ;**

**TextField txtValuta ;**

**DatePicker dpDataScadenta ;**

**TextField txtTipFactura ;**

**Button btnLogout ;**

**Button btnCancel ;**

**TextField txtCampanie ;**

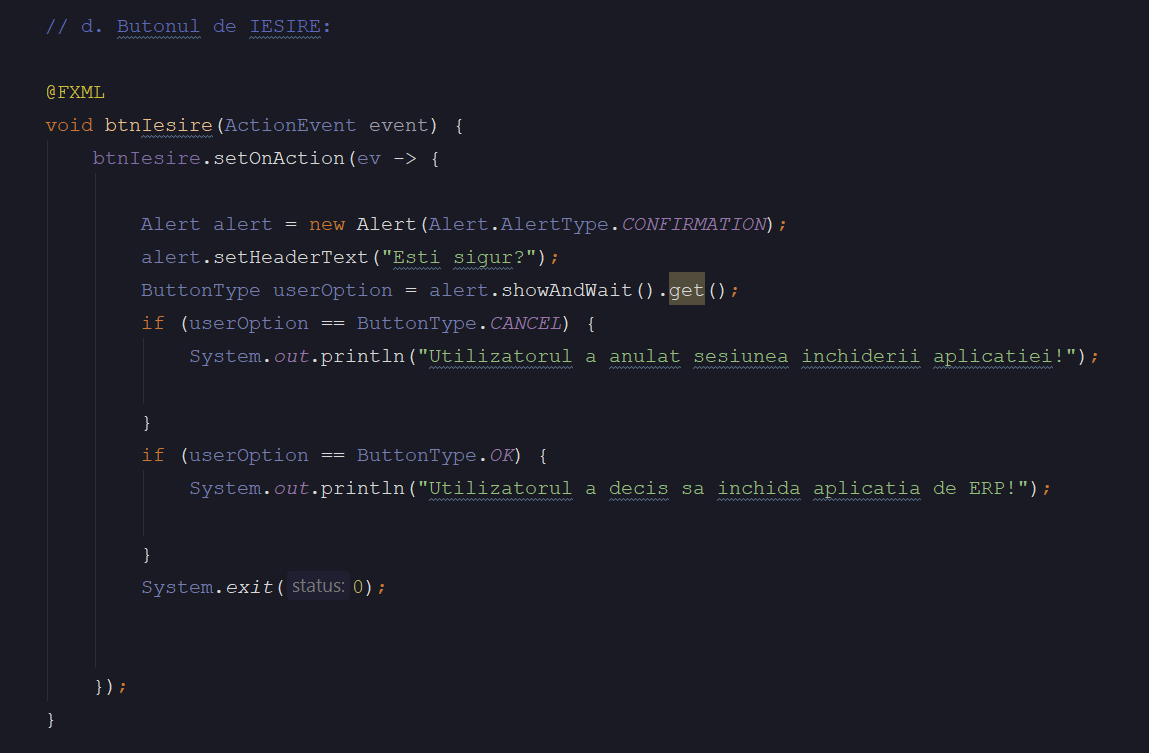
**Label lblTVA ;**

**Label lblSumacuTVA ;**

**TextField txtSumafaraTVA .**

* Funcționalitatea acestei clase ar fi următoarea :

1. Pentru **butonul de ieșire (btnIesire)**, modul de realizare este exact același ca și în celelalte clase prezentate :



1. Butonul **CANCEL** din acest forumular :

Cu ajutorul metodei **setText(null)**, setează tuturor variabilelor valoarea **null**.



1. Butonul de **LOGOUT** – ***cel de revenire la formularul de LOGIN*** :

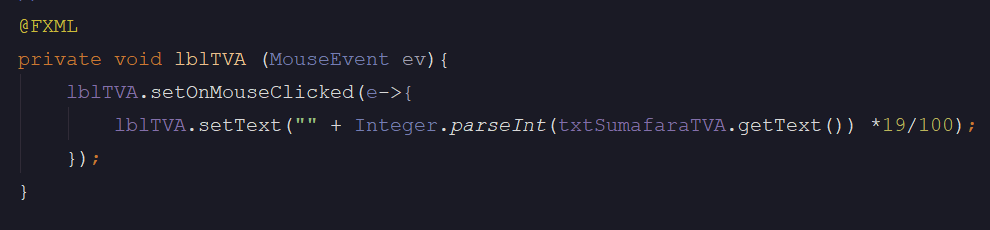
Acest buton a fost gândit în felul următor : după ce factura fiscală a fost completată sau mai multe facturi fiscale au fost înregistrate în baza de date, pentru a ieși din aplicație fie utilizezi butonul de ieșire (btnIesire) , fie apeși pe butonul de Logout (btnLogout) care te va da la ecranul principal.



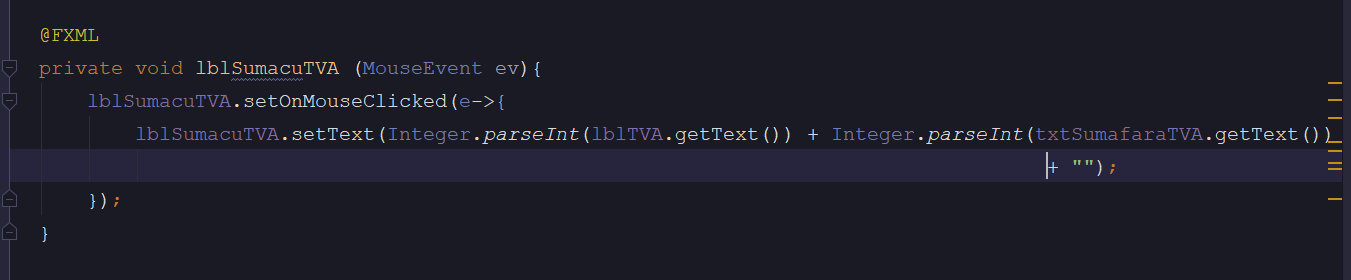
1. Cele **două label-uri** calculate din program:

Aceste două Label-uri apelează la evenimentul de tip ***MouseEvent***.

Pentru **Label-ul lblTVA** se va utiliza următorul cod în care suma fără TVA introduse de către utilizator, i se va extrage TVA-ul din suma respectivă. ***Cota de TVA*** utilizată în acest program este de **19%**.

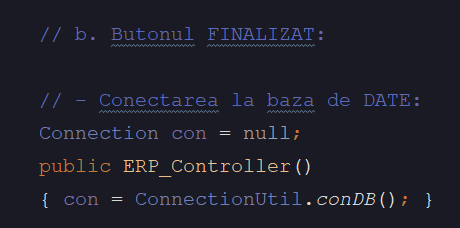


Pentru **Label-ul lblSumacuTVA** se va utiliza următorul cod în care sunt adunate cele două : TVA și SumafaraTVA.

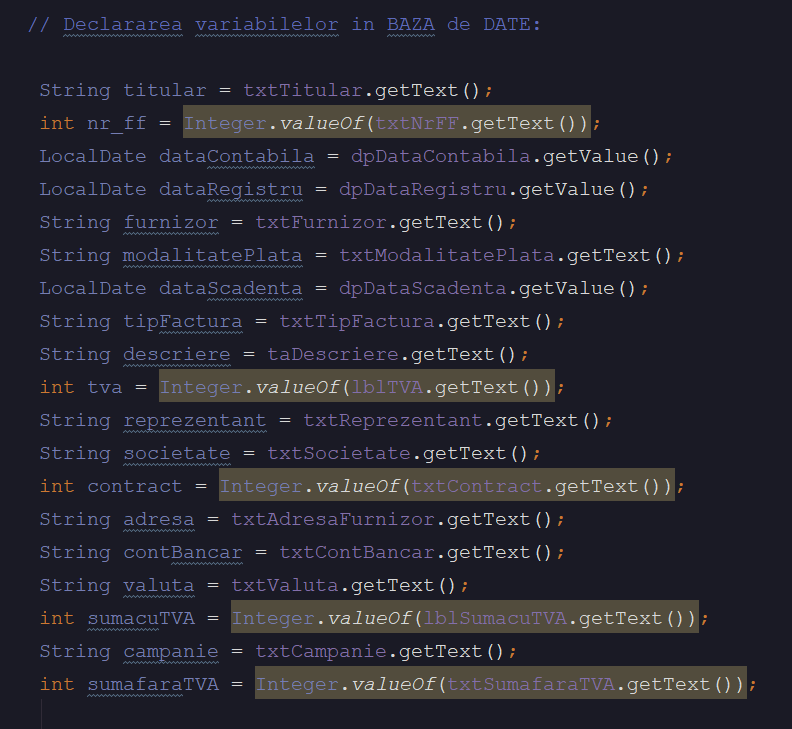


1. Butonul **FINALIZAT** – ***cel care specifică înregistrarea facturii fiscale*** :

Pentru început se face conexiunea la baza de date :



Pasul următor este apelarea evenimentului de tip ***ActionEvent***. Declarăm variabilele necesare exact așa cum au fost declarate și în baza de date :



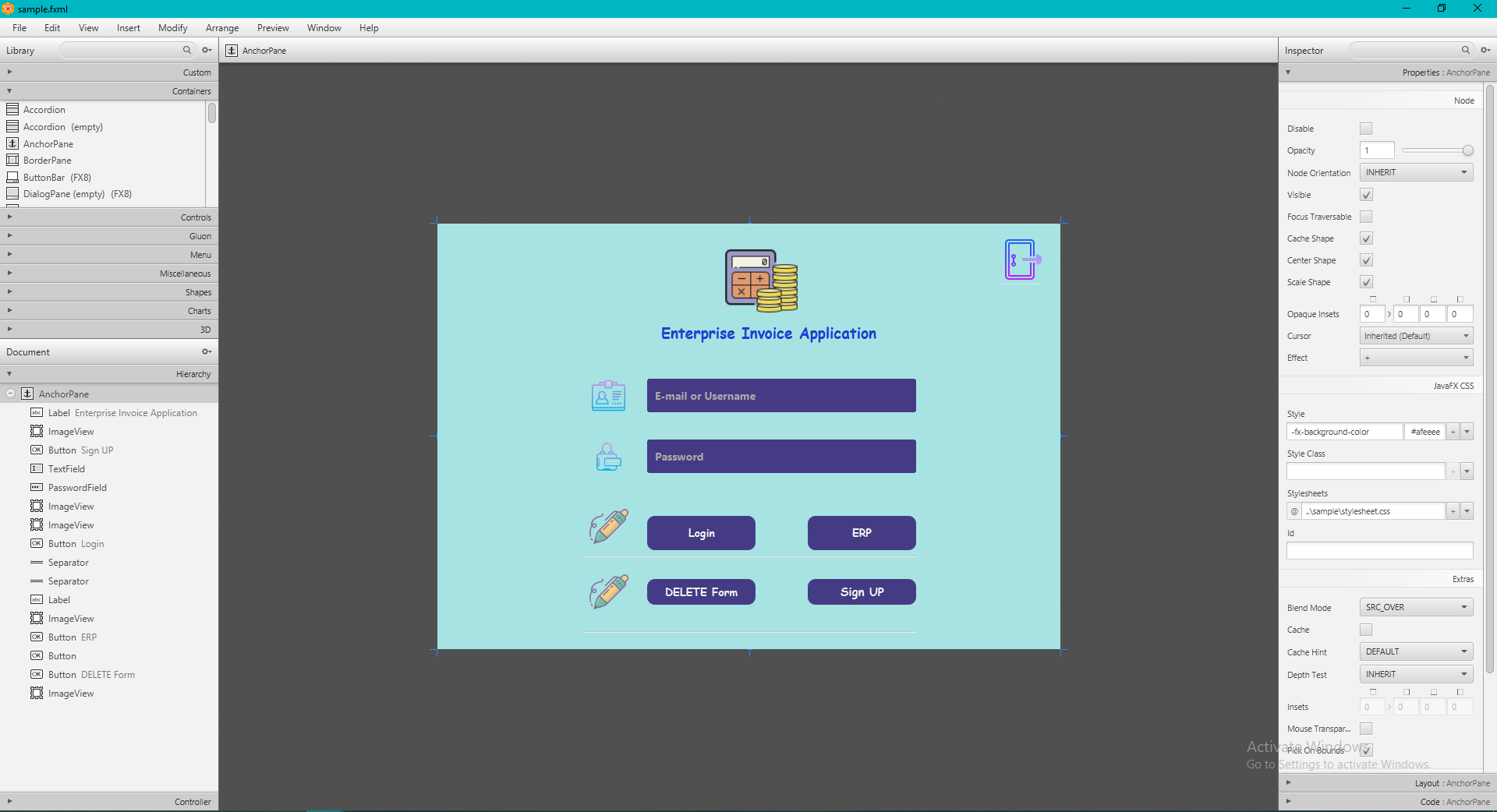
În blocul de instrucțiune **TRY...CATCH**, în partea de ***TRY***, la fel ca și în celelalte clase prezentate, pașii rămân la fel, doar că ***instrucțiunea SQL*** este alta :

**String sql = ”INSERT INTO erp (titular, nr\_ff, dataContabila, dataRegistru, furnizor, modalitatePlata, dataScadenta, tipFactura, descriere, tva, reprezentant, societate, contract, adresa, contBancar, valuta, sumacuTVA, campanie, sumafaraTVA) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) ” ;**

La fel ca și la celelalte clase prezentate, se va declara ca metoda ***preparedStatement.setString()*** modul în care variabilele au fost declarate în baza de date.

La final, se va face un ***AlertType*** de tip **INFORMARE**, cu acest mesaj: ”Factura completată cu succes!”.

1. **Partea de interfață cu utilizatorul :**
2. Formularul sample.fxml :



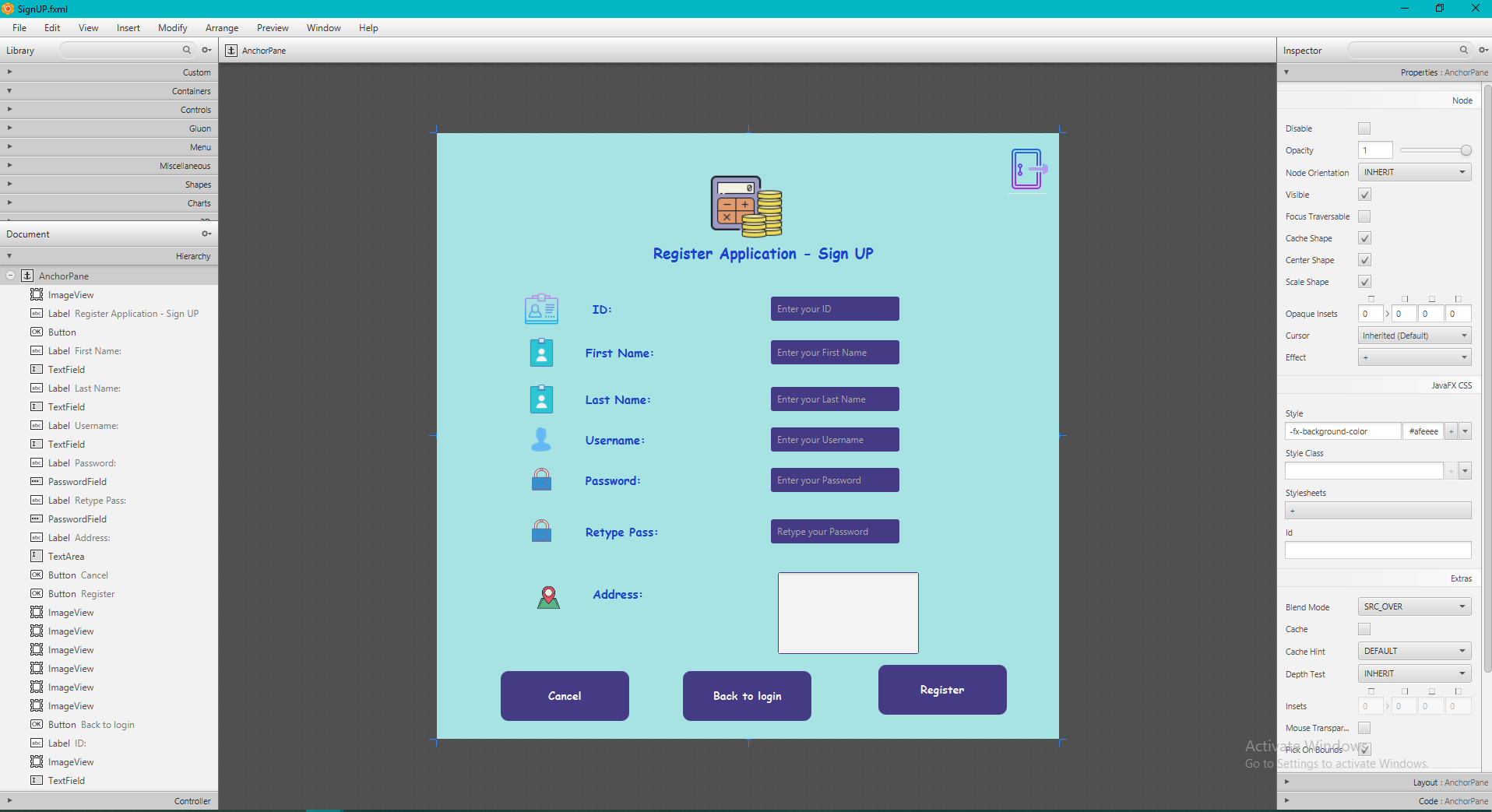
Conform acestei poze, în această interfață avem următoarele componente :

* 2 Label-uri ;
* 5 ImageView-uri ;
* 5 Buttons ( butoane) ;
* 1 TextField ;
* 1 PasswordField ;
* 1 AnchorPane ;
* 2 Separator .

Am ales ca în această interfață să utilizez și partea de JavaFX-CSS.

Astfel, pentru background (culoarea AnchorPane-ului de fapt), am ales culoarea: #afeeee. Pentru restul design-ului, am utilizat un fișier CSS numit : stylesheet.css ( partea de rotunjire a butoanelor). Pentru partea de background a TextField-ului, PasswordField-ului, respectiv celor 4 butoane (Login, ERP, Delete Form, Sign UP) am alees culoarea #483D8B.

1. Formularul de SignUP.fxml :



Conform acestei poze, în această interfață avem următoarele componente :

* 1 AnchorPane ;
* 8 ImageView-uri ;
* 8 Label-uri ;
* 4 Buttons ( butoane) ;
* 4 TextField-uri ;
* 2 PasswordField-uri ;
* 1 TextArea.

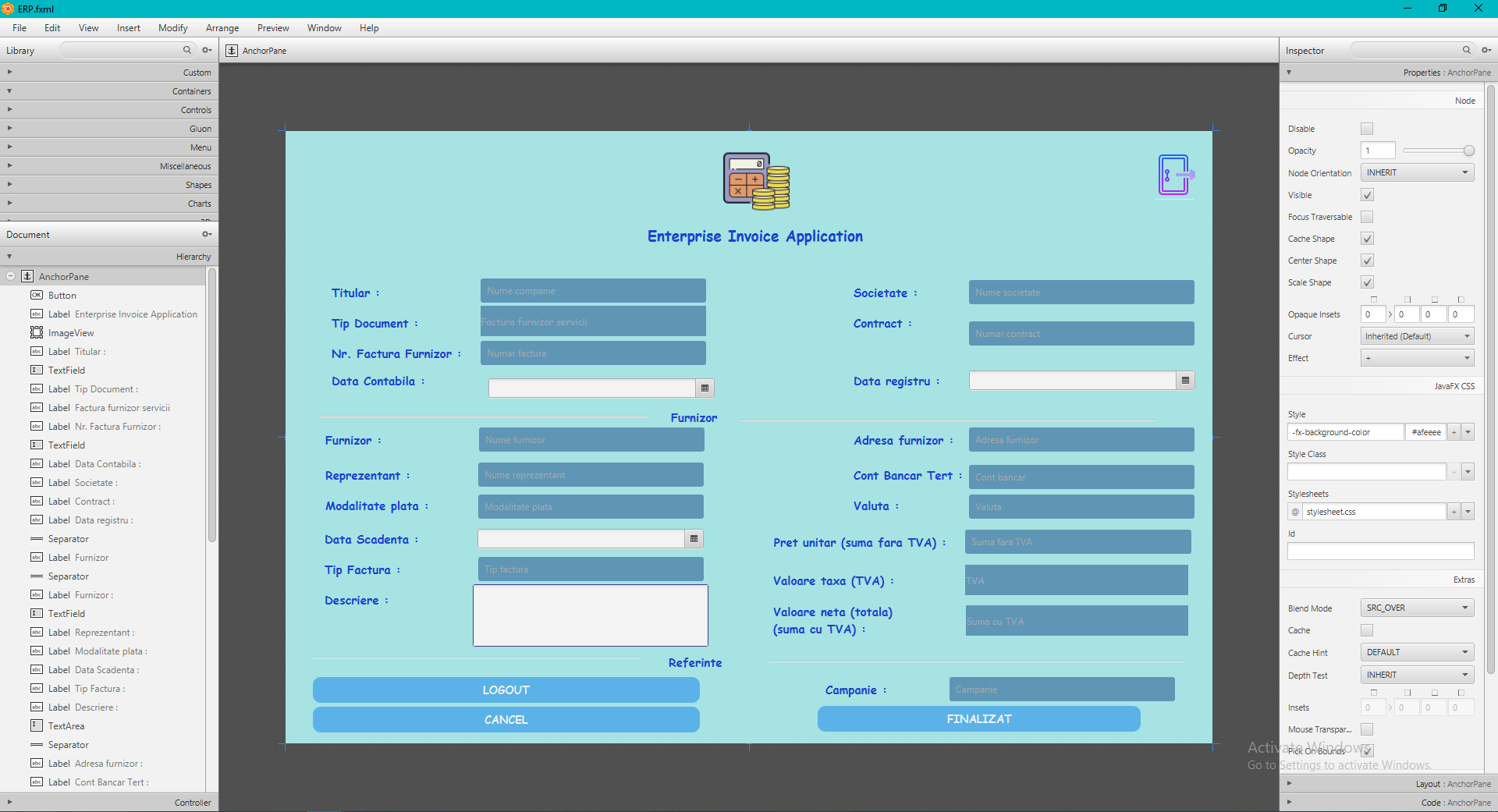
Am ales ca în această interfață să utilizez și partea de JavaFX-CSS.

Astfel, background-ul va fi același în toate formularele, culoarea fiind **#afeeee.**

Pentru restul design-ului, am utilizat un fișier CSS numit: stylesheet.css ( partea de rotunjire a butoanelor).

Pentru partea de background a componentelor din formular, am ales culoarea **#483D8B.**

1. Formularul de ERP.fxml :



Conform acestei poze, în această interfață avem următoarele componente:

* 1 AnchorPane ;
* 4 Buttons (butoane) ;
* 1 ImageView ;
* 26 Label-uri ;
* 12 TextField-uri ;
* 4 Separator;
* 1 TextArea ;
* 3 DatePicker-uri.

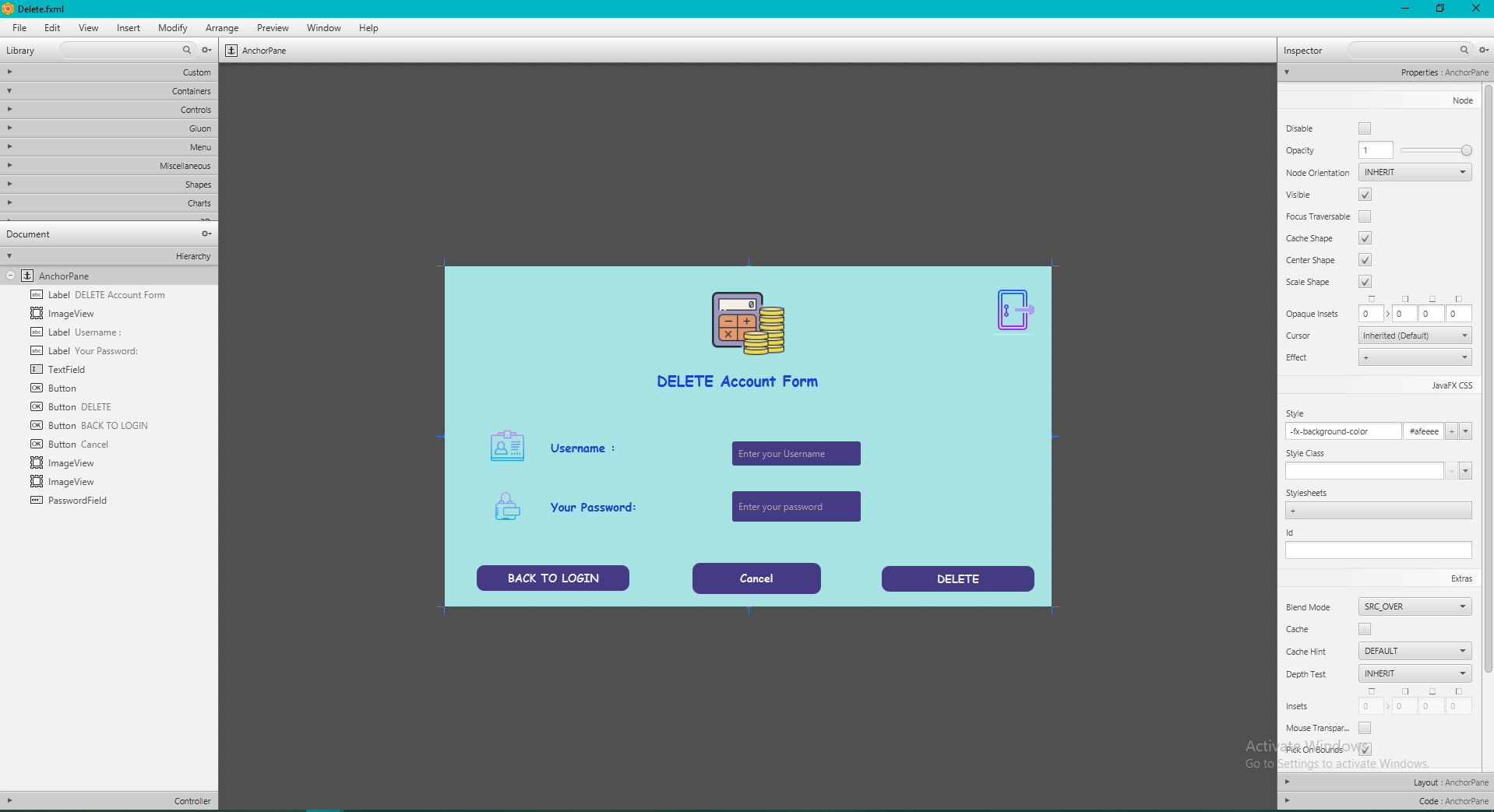
Am ales ca în această interfață să utilizez și partea de JavaFX-CSS.

Astfel, background-ul va fi același în toate formularele, culoarea fiind #afeeee.

Pentru restul design-ului , am utilizat un fișier CSS numit : stylesheet.css ( partea de rotunjire a butoanelor).

Pentru partea de background a componentelor din formular, am ales culoarea #649bdf; culoarea butoanelor din formular fiind #5ebbf4.

1. Formularul DELETE :



Conform acestei poze, în această interfață avem următoarele componente :

* 1 AnchorPane ;
* 3 Label-uri ;
* 3 ImageView-uri ;
* 1 TextField ;
* 4 Buttons (butoane) ;
* 1 PasswordField .

Am ales ca în acestă interfață să utilizez și partea JavaFX-CSS.

Astfel, background-ul va fi același în toate formularele, culoarea fiind #afeeee.

Pentru restul design-ului, am utilizat un fișier CSS numit: stylesheet.css ( partea de rotunjire a butoanelor).

Pentru partea de background a componentelor din formular, am ales culoarea #483D8B.

**CAPITOLUL III – Concluzii:**

Pot spune că această aplicație poate fi utilizată de către oricine chiar și de către oamenii de rând, nu doar de contabili. Și știți de ce? Pentru că aplicația poate ține evidența facturilor de zi cu zi. Așa este mai ușor de utilizat de către oricine.

Fiind o aplicație la început de drum, și încă nu poate fi comparată cu ce este pe piață, această aplicație se mai poate dezvolta și anume îi pot fi adăugate pe viitor următoarele module :

* modulul de bancă ;
* modulul de casă ;
* modulul de comenzi ;
* modulul de gestiune ;
* modulul de ofertare ș.a. .

Ca și părți negative, pot spune că această aplicație fiind la început, nu are și o capacitate de a fi securizată propriu-zis, dar pe viitor pot fi adăugate.

În concluzie, aplicația Enterprise Invoice Application este destinată oricui și poate fi adresată oricărui om ce dorește să aibă în evidență facturile venite pentru o posibilă plată.