PROIECT MDS
GRUPA 232

COMPANIE DE TRANSPORT

Profesori:
Alin Stefanescu
Mihai Prunescu

Autori:
RAUL GIOANCA
LUIZA ACHIM

Cuprins

1 Introducere **2**

2 Prezentarea aplicatiei **4**

3 Codul Sursa **13**

4 Gestionarea datelor **15**

5 Design, arhitectura **18**

6 Procesul de creatie **20**

7 Principalele probleme intalnite **22**

8 Source control **25**

9 Teste automate **27**

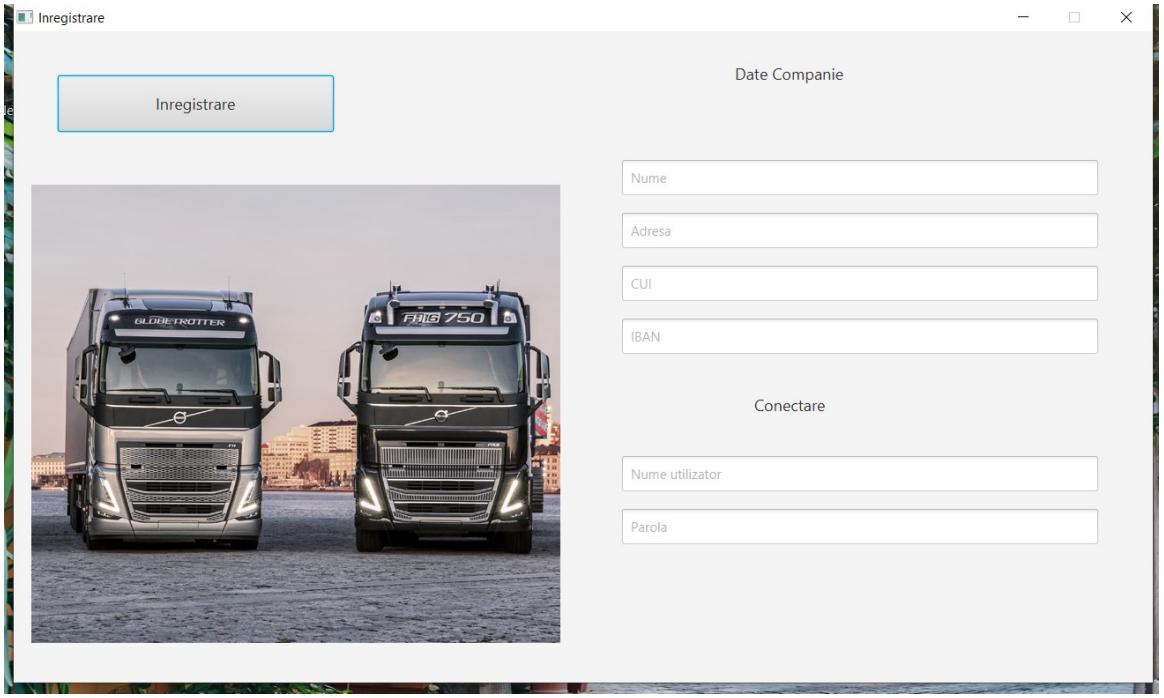
10 Incheiere **29**

1 Introducere

Aceasta reprezinta documentatia proiectului ce poate fi gasit la adresa: `https://github.com/Raul-D-G/LKW`.

Proiectul a fost creat folosind limbajul Java si ofera o interfata grafica, cu ecrane multiple, ce poate fi folosita pentru gestionarea unei companii de transport. Implementeaza actiuni multiple, avand de asemenea un design minimalist. Nu prezinta ambiguitate, pentru a sustine o interactiune cu utilizatorul simpla si eficienta. Are o constructie robusta, ce ofera siguranta pastrarii datelor si maleabilitate in prelucrarea lor.

Produsul final este un succes. Inca de la prima vedere, prezinta un potential urias de a fi utilizat in viata reala, fiind capabil sa se muleze in functie de cerintele clientilor. Reprezinta o unealta ce poate ajuta la organizarea compacta si facila a unei baze de date si sustine dezvoltarea continua a companiei in cauza.

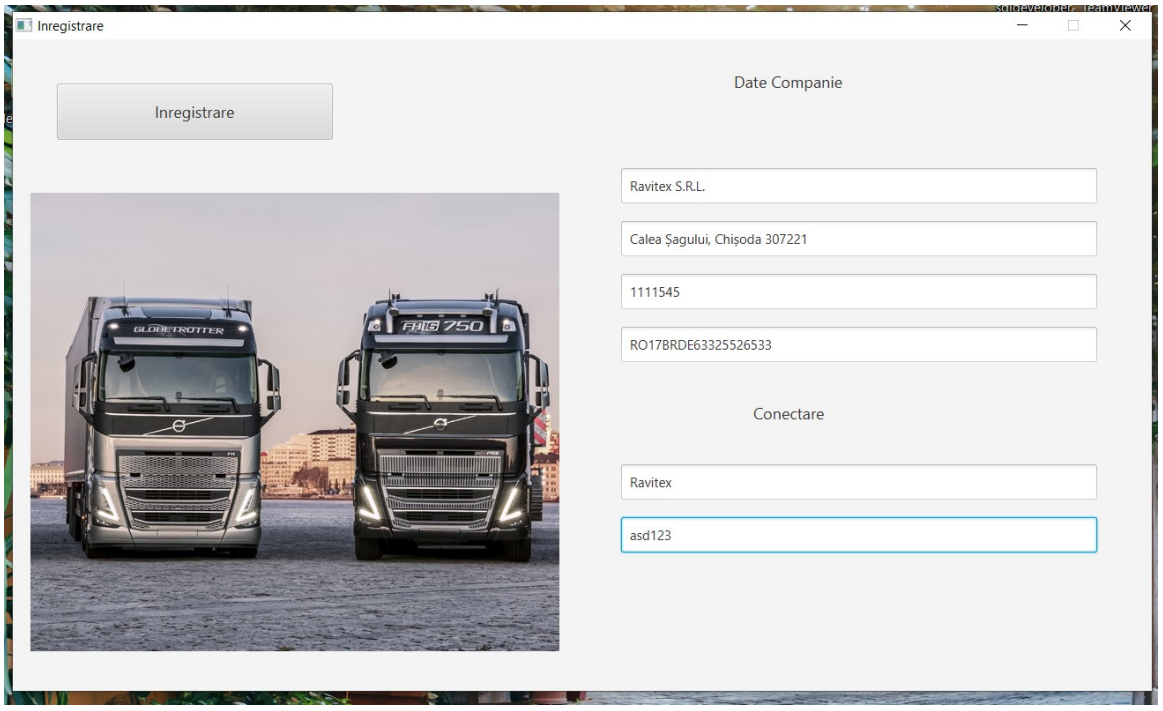


Motivatia ce a stat la baza crearii acestui proiect a pornit desigur de la tema din cadrul cursului "Metode de Dezvoltare Software ". In momentul in care echipa s a intalnit pentru prima oara pentru alegerea temei, asa zisul proces de "Brain Storming" am convenit la o cu totul alta tema pentru aplicatie.
Ulterior, decizia noastra s a schimbat deoarece am ajuns la concluzia ca este mult mai potrivit un proiect cu aplicabilitate in viata reala, decat unul care doar replica niste aspecte imaginative care ne incanta personal.

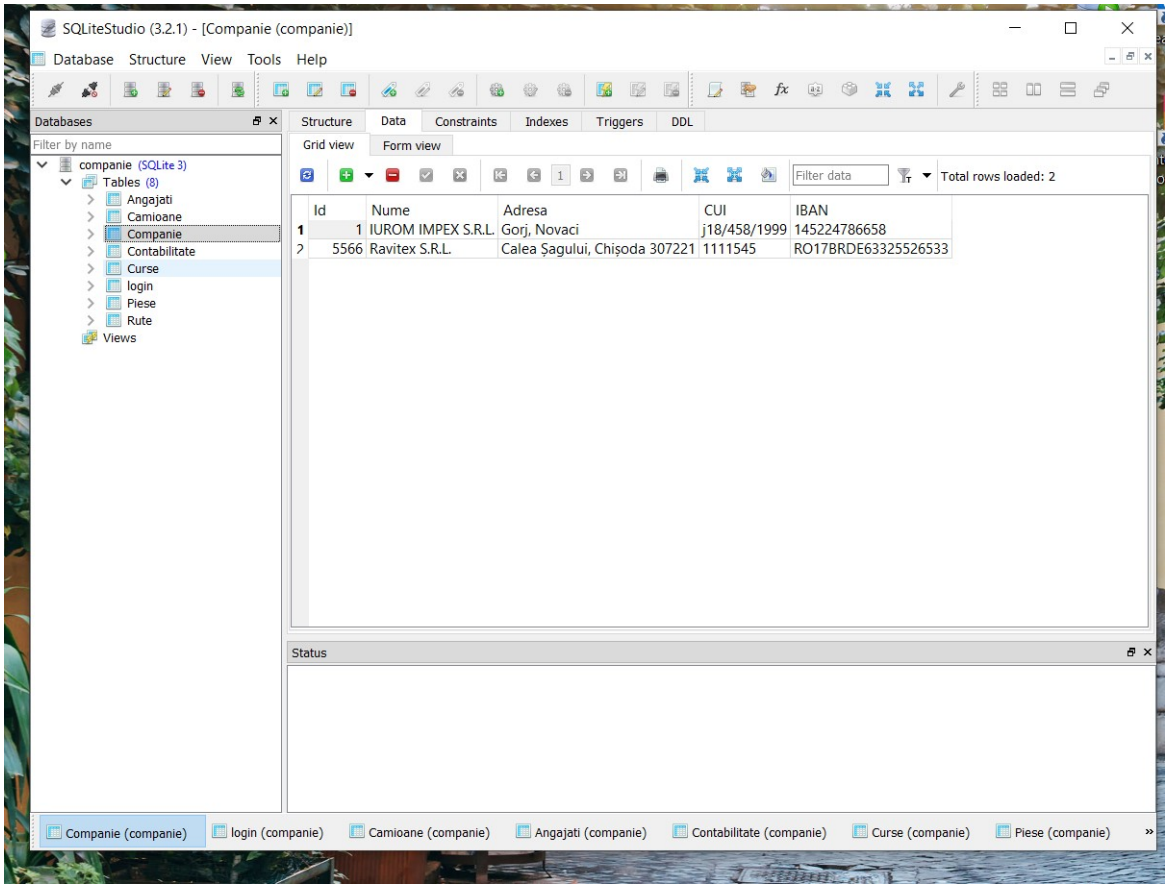
Astfel am ajuns sa impersonam proprietarii unei companii de transport, care au disperata nevoie de un sistem de a gestiona cursele, angajatii si bunurile. Trebuia sa aiba o interfata grafica care sa satisfaca nevoia de simplitate si coerenta a utilizatorilor. Asa a luat nastere proiectul cu numele sau de cod "LKW".

2 Prezentarea aplicatiei

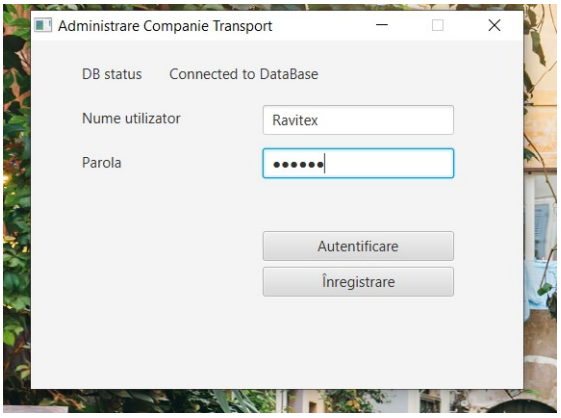
Cel mai bun mod de a prezenta aplicatia este de a oferi o simulare de inregistrare si de valorificare a functionalitatilor oferte.
Consideram ca impersonam un utilizator nou care se autentifica pentru prima oara in aplicatie. Completam campurile corespunzator: nume, adresa, cod unic de inregistrare, IBAN; respectiv datele pentru login: username, parola.



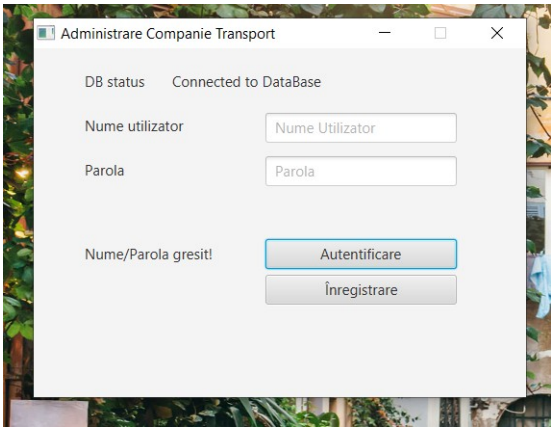
Odata ce ne am autentificat, se creaza o intrare in baza de date cu credentialele folosite.



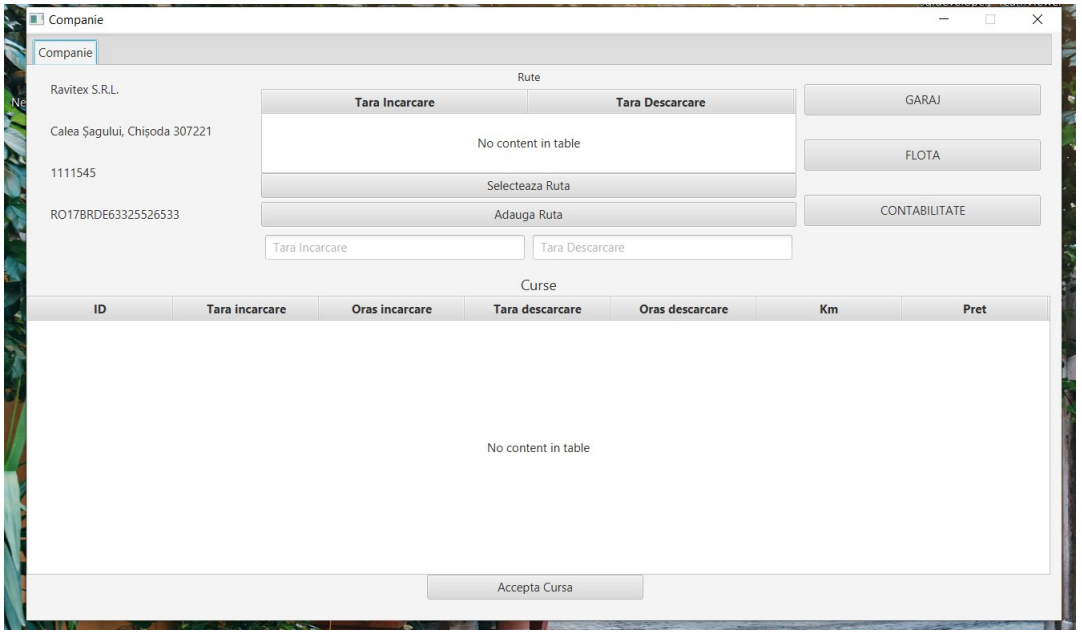
Acum putem sa revenim la meniul anterior si sa ne autentificam in mod corespunzator.



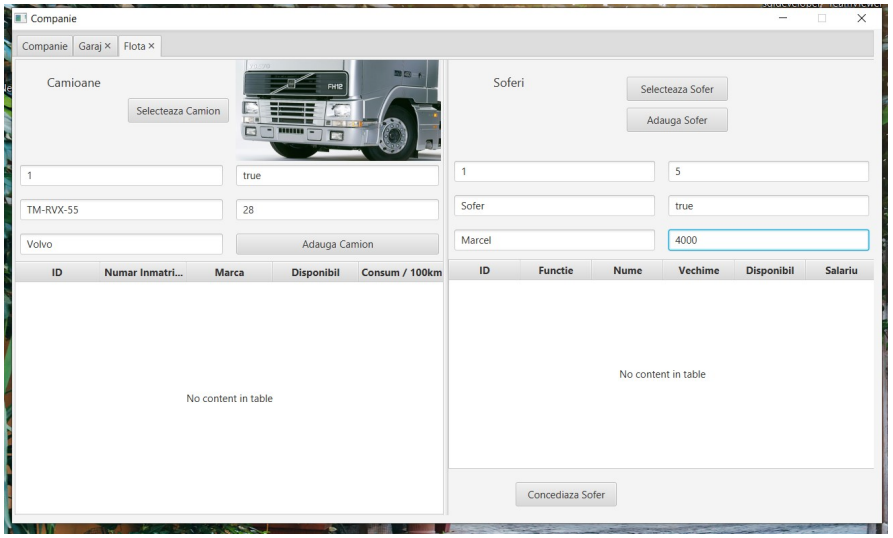
Daca gresim datele de conectare (difera fata de cele din baza de date) atunci vom fi atentionati printr un mesaj de eroare ”Nume/Parola gresite!” si vom ramane pe ecranul principal.

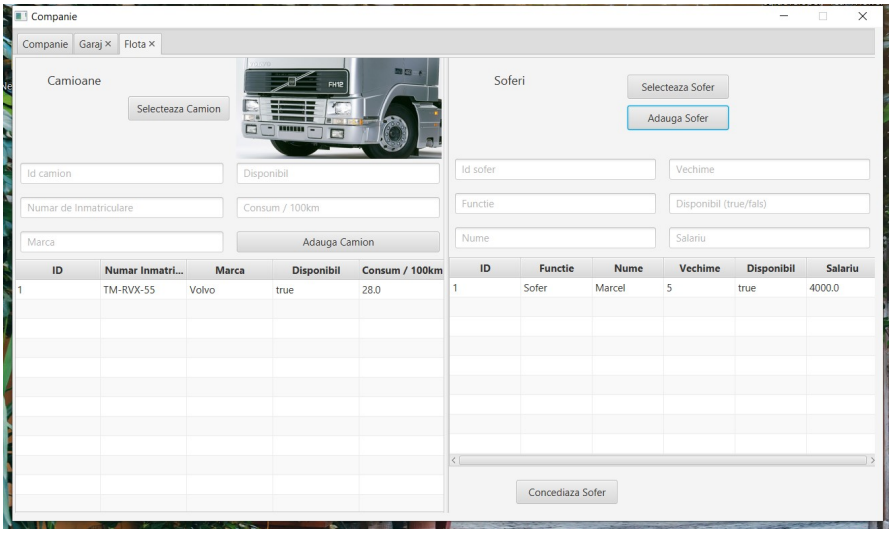


Daca autentificarea se face cu succes (nu gresim username si parola) atunci vom fi condusi la urmatorul ecran, care surpinde detaliile despre companie.

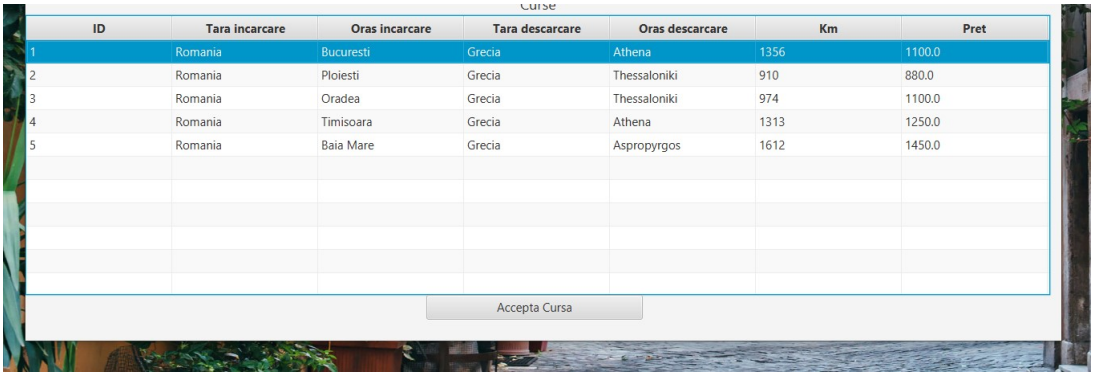


Compania noastra nu are insa momentan niciun bun in posesie. Acest lucru poate fi verificat intrand in meniul "Flota".
Deoarece dorim o simulare completa a procesului, scopul nostru este sa acceptam o cursa pentru a o adauga in contabilitate. Pentru aceasta avem nevoie de cel putin un camion si un sofer disponibili (care sa aiba campul Disponibil setat cu true; daca campul este false sau nu exista inregistrare, atunci nu vom putea accepta cursa)

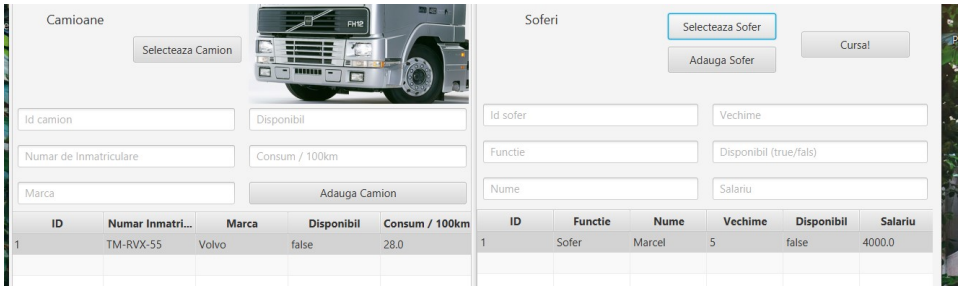




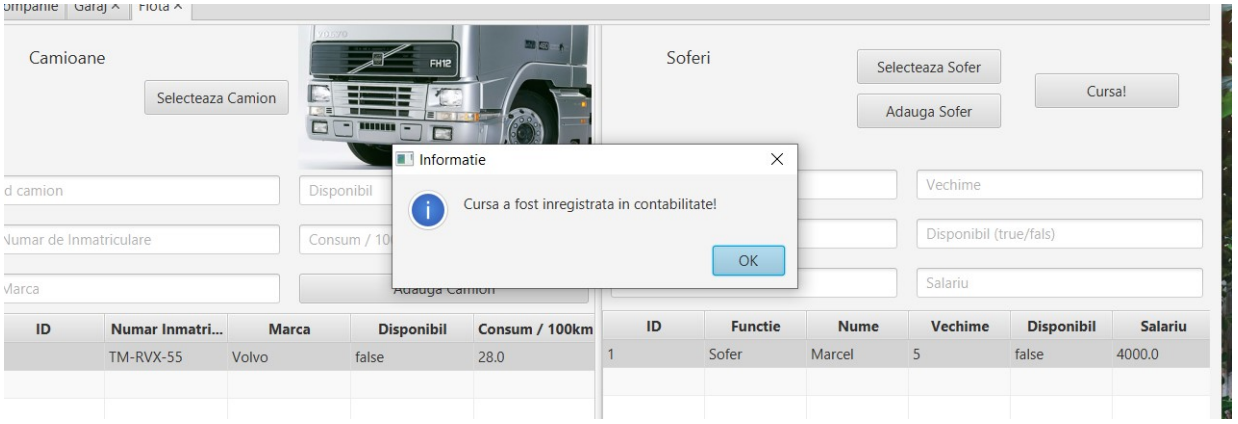
Acum ca ne am asigurat ca avem tot ce ne trebuie pentru a accepta o cursa, ne intoarcem in meniul "Companie", si alegem o ruta din cele disponibile (ordinea butoanelor este fireasca: Selecteaza Ruta $\longrightarrow$ Accepta Cursa).



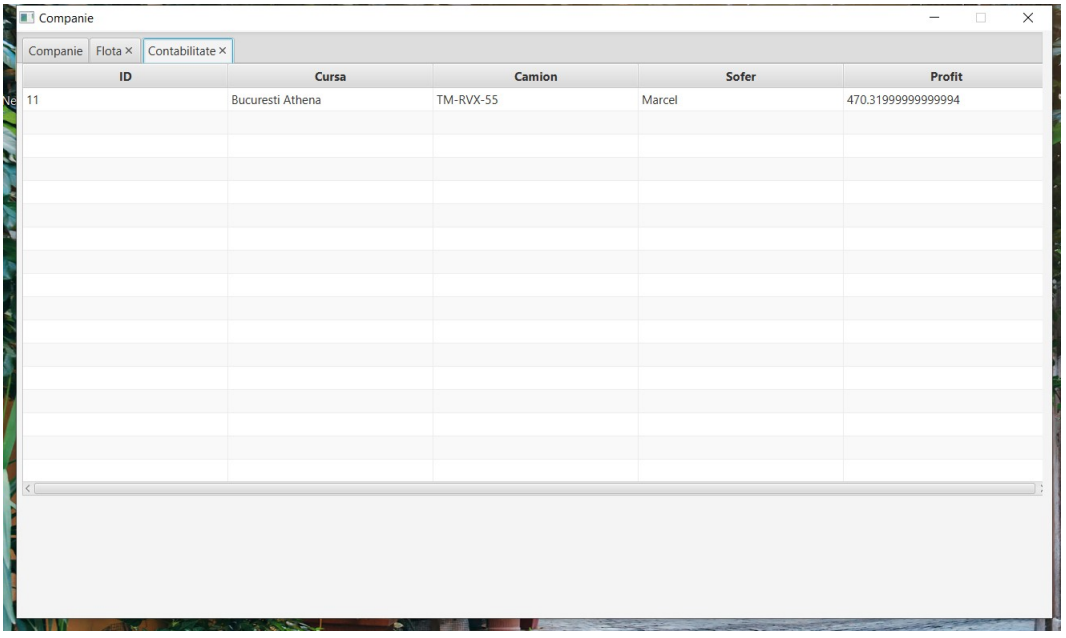
Suntem condusi acum in meniul ce contine resursele necesare, urmand sa selectam un camion si un sofer disponibili. Actiunea este finalizata prin apasarea butonului "Cursa!" care apare doar in situatia relatata(cand dorim sa preluam o noua cursa).



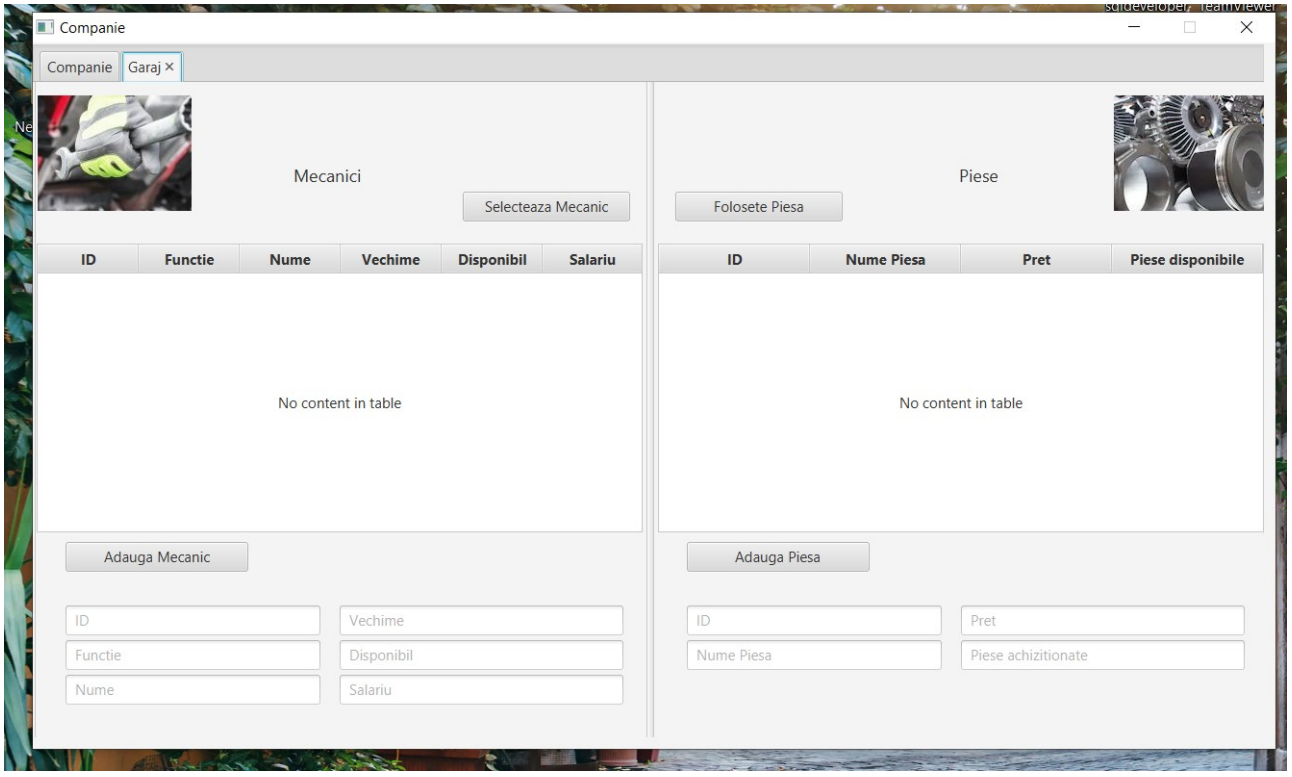
Dupa ce procesul a fost finalizat cu succes, pe ecran va aparea un mesaj corespunzator care ne anunta ca a fost inregistrata cursa.



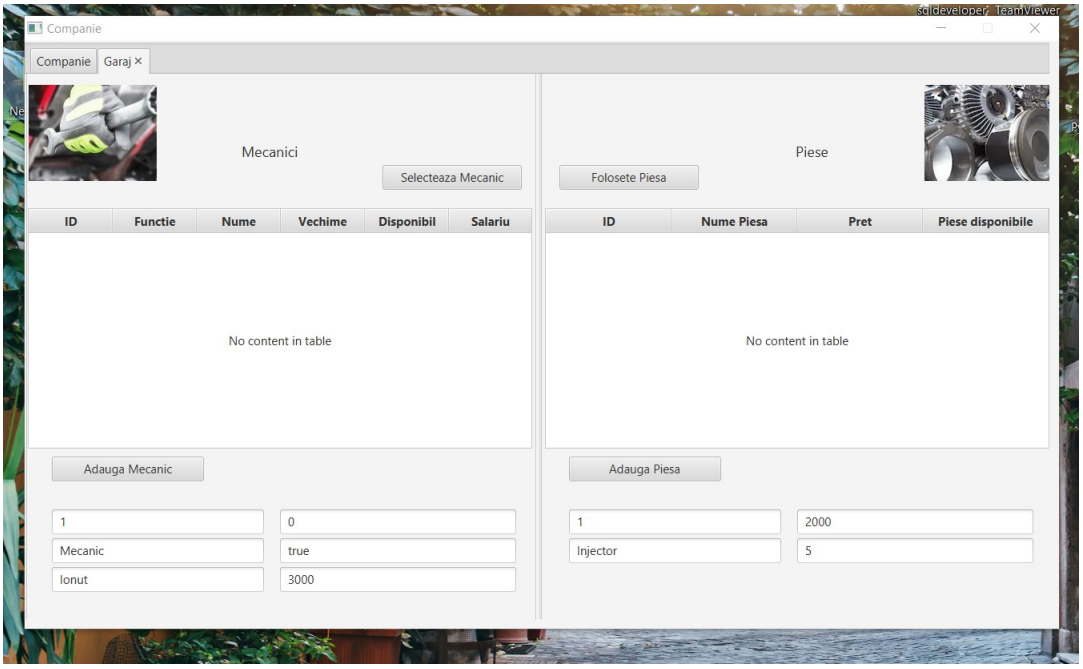
Pentru a verifica acest lucru, putem acum sa intram in meniul ”Contabilitate” unde vom vizualiza cursa.

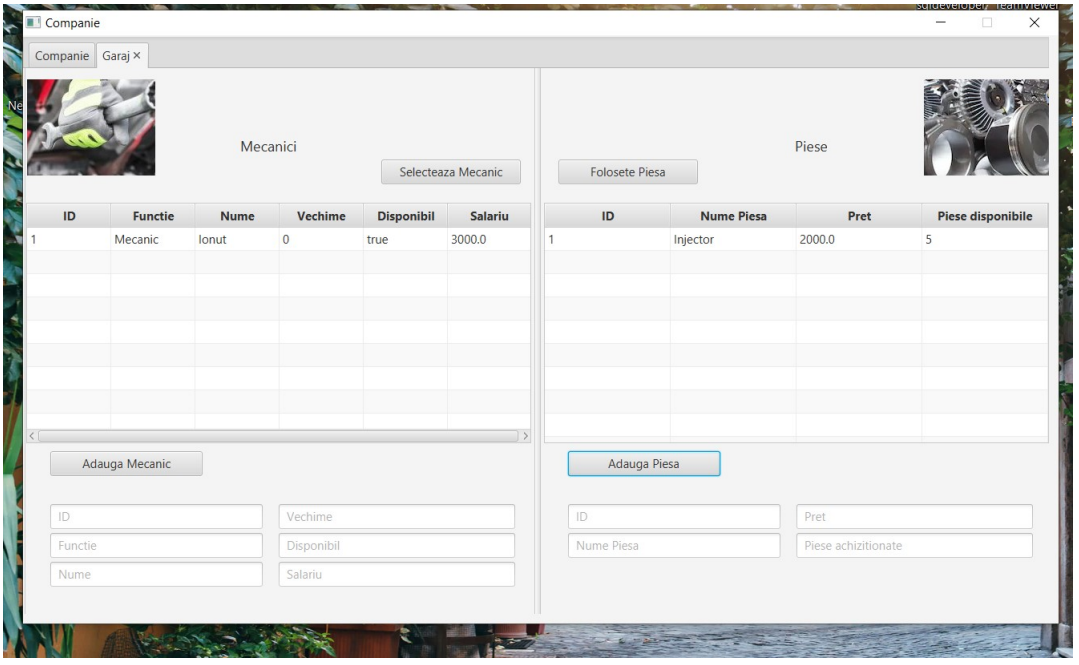


Aplicatia ofera insa si functionalitatea de ” inventar” unde putem tine socoteala angajatilor externi(mecanici) si a pieselor, fiecare cu speficiatiile sale.



Ca si exemplu didactic, vom crea cate o inregistrare pentru fiecare categorie. De asemenea, exista campuri specifice care contorizeaza disponibilitatea bunurilor(true,false pentru mecanici, respectiv un numar de piese disponibile pentru fiecare inregistrare). Daca selectam o piesa, atunci vom scadea 1 din numarul pieselor disponibile.


Companie
Companie
Garaj ×
Mecanici
Selecteaza Mecanic
Folosete Piesa
Piese
ID
Functie
Nume
Vechime
Disponibil
Salariu
No content in table
ID
Nume Piesa
Pret
Piese disponibile
No content in table
Adauga Mecanic
Adauga Piesa
1
0
Mecanic
true
Ionut
3000
1
2000
Injector
5

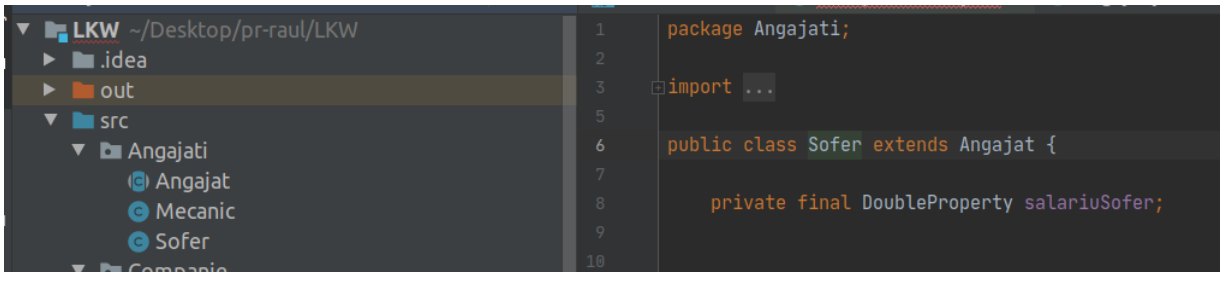

Companie
Companie
Garaj ×
Mecanici
Selecteaza Mecanic
Folosete Piesa
Piese
ID
Functie
Nume
Vechime
Disponibil
Salariu
1
Mecanic
Ionut
0
true
3000.0
ID
Nume Piesa
Pret
Piese disponibile
1
Injector
2000.0
5
Adauga Mecanic
Adauga Piesa
ID
Vechime
Functie
Disponibil
Nume
Salariu
ID
Pret
Nume Piesa
Piese achizitionate

Acum ne putem intoarce in meniul "Companie" si jongla cu aceste functii disponibile. Putem sa cream un numar infinit de inregistrari de orice fel, care vor fi stocate pentru noi, in siguranta.
O imbunatatire care va avea prioritate in viitoarea dezvoltare a aplicatiei o reprezinta butonul de "Log out". De asemenea, avem in vedere adaugarea unui nou tabel pentru tipul de marfa transportata si detaliile despre aceasta.
Pentru moment, va incurajam sa va obijnuiti cu schema incipienta a aplicatiei prin testarea unor exemple similare cu cele prezentate anterior.

3 Codul Sursa

Pentru munca investita efectiv in cod, la nivel de echipa am convenit sa folosim IntelliJ. Intrucat acesta este unul dintre cele mai folosite IDE uri pentru dezolvtare Java, am ales sa jucam la sigur si sa nu ne expunem la niciun fel de diferente de perspetiva.

Codul sursa consta in aproximativ 10 pachete, organizate metodic in functie de ceea ce contin, alaturi de fisiere auxiliare de diverse tipuri(.png, .jpeg, .sqlite, .pdf, .md). Fiecare pachet contine doar clase care impart anumite functionalitati, care se extind sau se implementeaza una pe cealalta. In acest mod pastram o ”curatenie” atat de necesara in cod.



Functi main se gaseste in pachetul ”LoginApp” in clasa cu acelasi nume. Este singura functie main din aplicatie si este cea care trebui rulata pentru a porni si testa aplicatia.

```
1   package LoginApp;
2
3   import ...
8
9   public class LoginApp extends Application {
10
11      public void start(Stage stage) throws Exception {
12          Parent root = (Parent) FXMLLoader.load(getClass().getResource( s: "login.fxml"));
13
14          Scene scene = new Scene(root);
15          stage.setScene(scene);
16          stage.setTitle("Administrare Companie Transport");
17          stage.setResizable(false);
18          stage.show();
19      }
20
21      public static void main(String[] args) { launch(args); }
```

De cate ori a fost nevoie, am introdus in cod metode specifice de tratare a erorilor. In acest fel ne asiguram ca o eroare este, in primul rand, semnalata, iar ulterior tratata ca atare.

```
}
catch (IOException ex) {
    ex.printStackTrace();
}
```

4 Gestionarea datelor

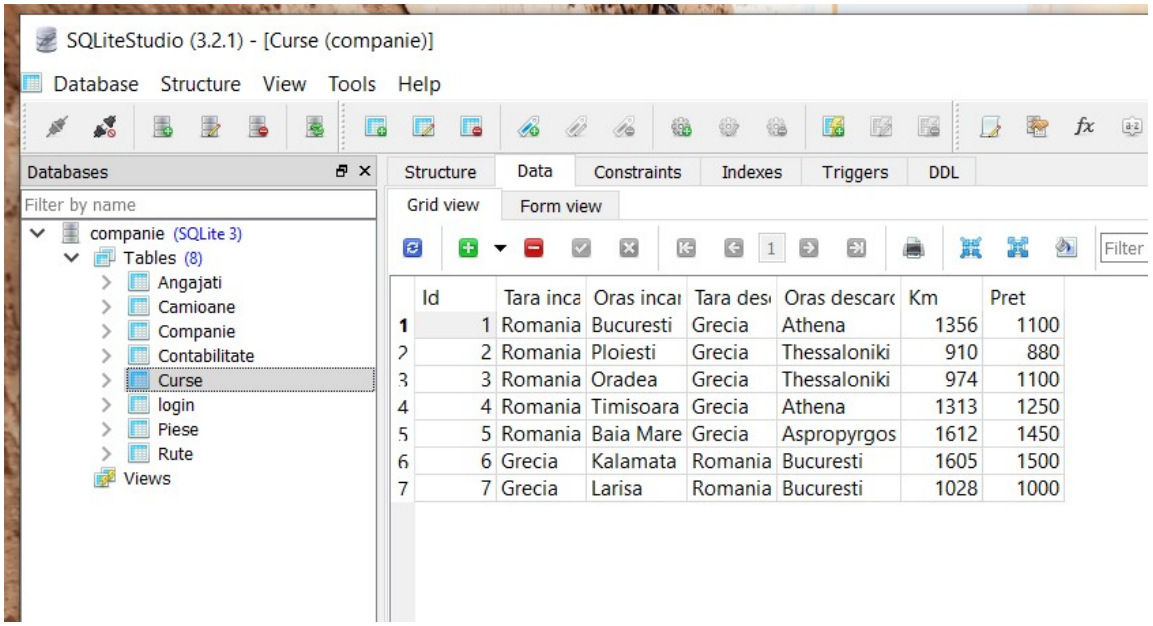
Intrucat aceasta este o aplicatie care, la nevoie, va trebui sa stocheze cantitati mari de date, am integrat in proiectul Java metode ce asigura persistenta datelor.
Am creat o conexiune folosind JDBC la o baza de date creata cu ajutorul SQLite(o versiune realtiv mai usor de folosit, care ne a facilitat crearea bazei de date) si am vizualizat o prin SQLiteStudio pentru a putea gestiona evetualele erori de citire, scriere.

```
package DbUtil;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DbConnection {

//    private static final String USERNAME = "dbuser";
//    private static final String PASSWORD = "dbpassword";
    private static final String SQCONN = "jdbc:sqlite:src/companie.sqlite";

    public static Connection getConnection() throws SQLException {

        try {
            Class.forName("org.sqlite.JDBC");
            return DriverManager.getConnection(SQCONN);
        }
        catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        return null;
    }
}
```

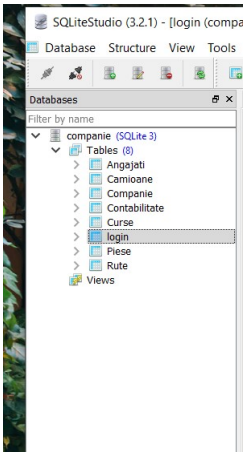
Unele date au fost introduse de mana prin SQLiteStudio pentru a avea o privire mai buna de ansamblu asupra structurii unui tabel nou creat. Alte date au fost introduce din cod si mai apoi sterse, aceasta metoda fiind ulterior inlocuita de metode specifice care realizeaza conexiunea la baza de date si transpun, prin cod, comenzile sql dorite.

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    String sql = "SELECT * FROM Contabilitate WHERE IdCompanie = ?";
    DbConnection dc = new DbConnection();
    int id;
    String cursa, camion, sofer;
    double profit;

    try {
        Connection conn = DbConnection.getConnection();
        assert conn != null;
        PreparedStatement pr = conn.prepareStatement(sql);
        pr.setInt( i: 1,LoginController.idCompanie);
        ResultSet rs = pr.executeQuery();

        ObservableList<Contabilitate> contabilitateObservableList = FXCollections.observableArrayList();

        while (rs.next()) {
```

Initial, baza de date contine anumite inregistrari predefinite pentru a putea testa unele functionalitati.

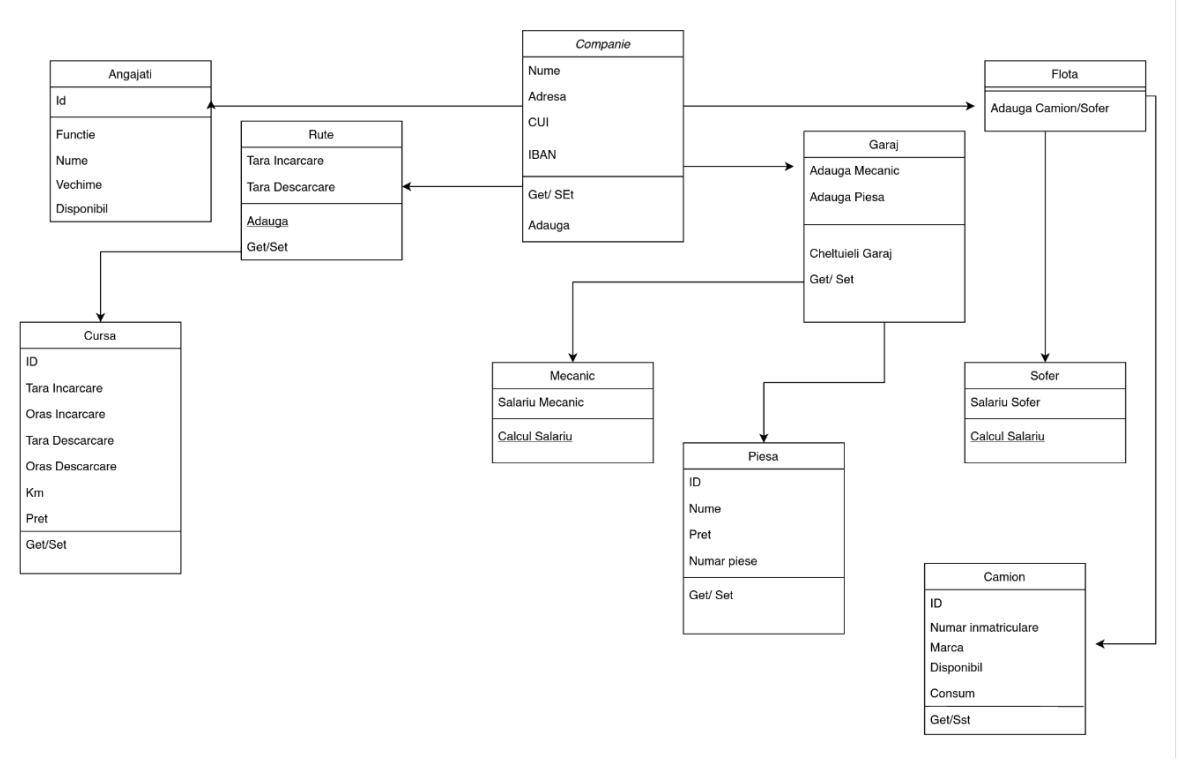


Tabelul curse este singurul care momentan nu are cum sa fie modificat (sa se adauge, updateze intrari in tabel) astfel ca el va fi populat deja cu anumite intrari.
In total avem 8 tabele, fiecare repezentand totalitatea inregistrarilor a ceea ce se specifica in denumire. Nu avem constrangeri pentru campurile de date(de genul cheie primara, cheie straina) astfel ca schema este una extrem de simpla.

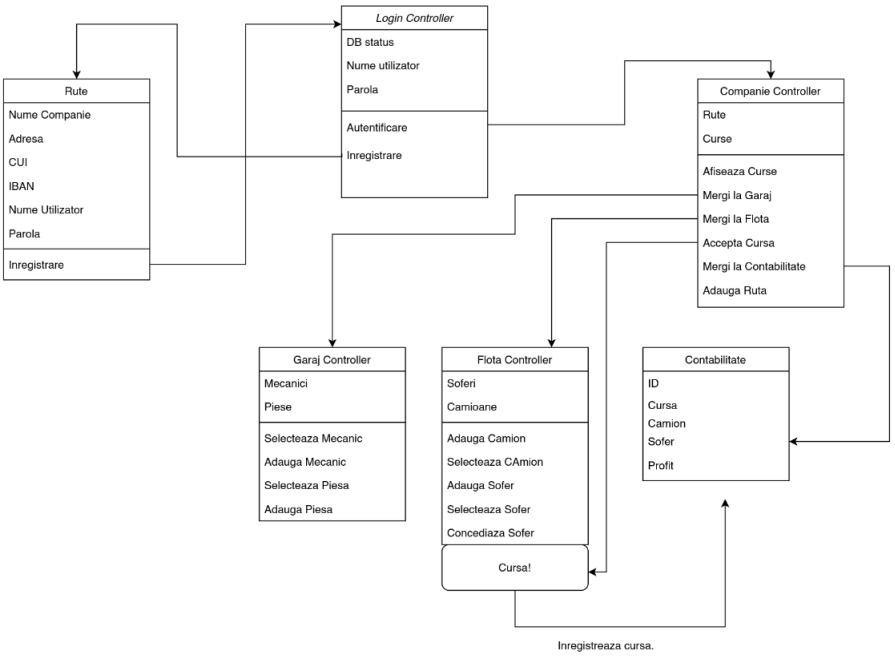


5 Design, arhitectura

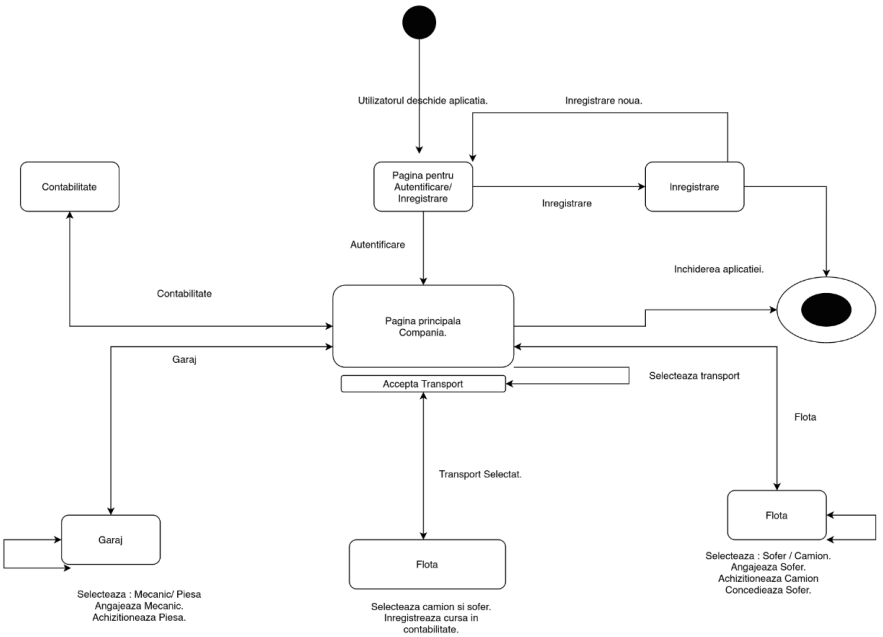
Pentru o mai buna intelegere a sistemului si analiza proprietatilor sale am create reprezentari grafice pentru anumite aspecte ale aplicatiei. In primul rand, avem definita structura claselor principale:



In al doilea rand, avem definita structura claselor de tip *Controller*. Aceste sunt adaugate ulterior, cand procesul de dezvoltare a aplicatiei ajunsese in punctul in care trebuia sa cream o legatura stabila intre date si interfata grafica. Astfel, clasele de tip controller controleaza flow ul de date si se asigura ca totul este afisat conform ultimului update .

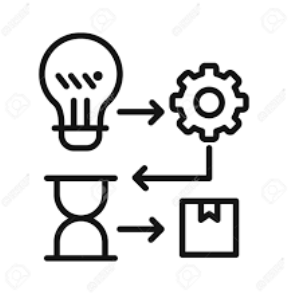


De asemenea, nu putea sa lipseasca o schematizare grafica a interactiunii untilizatorului cu aplicatia pentru clarificarea completa a tuturor detaliilor.

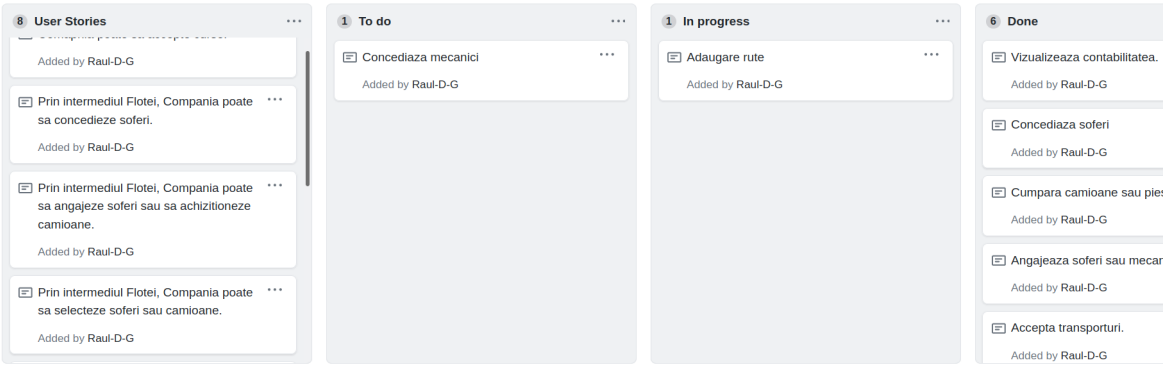


6 Procesul de creatie

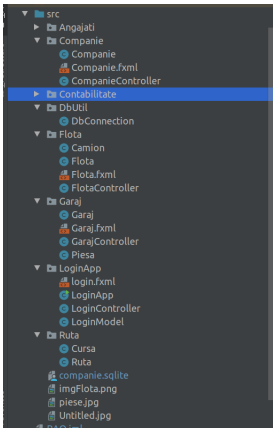
Desi produsul final nu are o interfata spectaculoasa si care sa te duca cu gandul la mii de ore de munca tehnica, dezvoltarea aplicatiei a constat intr-un lung sir de adaugari, imbunatatiri, reparatii, readaptari si poate chiar dizolvarea unor clase intregi.



Am decis sa mergem intr un ritm controlat si urmand anumiti pasi prestabiliti:
⟶ Stabilirea obiectivului(Ex:Compania sa poata accepta curse)
⟶ Scriere codului
⟶ Corectarea evetualelor eorori
⟶ Testarea efectiva a functionalitatii pe un exemplu aleatoriu
Acest lucru a fost desigur surprins, pentru a oferi dovada posteritatii asupra modului in care s a efectuat lucrul.
Pentru eficinta, am profitat de anumite facilitati oferite de platforma Github. Am decis ca un mediu ordonata este singurul mod in care putem duce la bun sfarsit un proiect de echipa si am actionat in consecinta.

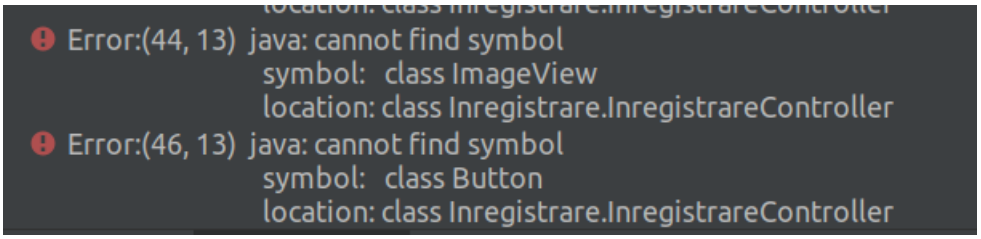


Codul este organizat astfel incat sa ofere o perspectiva aerisita, modularizata si usor de urmarit de o persoana care nu a fost implicata in procesul de creatie.



7 Principalele probleme intalnite

Drumul pana la rezultatul dorit nu a fost unul lin. Ne am confruntat cu un numar mare de erori, cele mai multe nesemnificative si care au fost corectate pe moment prin simpla revizuire a codului.



Deoarece proiectul are insa dimensiuni destul de mari, am incercat sa nu lasam nicio eroare netrata mai mult de cateva ore, intrucat daca am fi continuat sa dezvoltam pe langa ele, am fi avut mult mai mult de suferit. Astfel, procesul de dezvoltare efectiva a aplicatie a fost probabil dublat de timpul alocat depanarii.
Au fost insa si momente care au blocat procesul evolutiv al aplicatiei in anumite puncte cheie de care parea ca nu vom putea trece. Avem chiar si cateva exemple de "Bug Report" pentru a putea contura mai bine o imagine a tuturor problemelor cu care ne am confruntat.

Bug report - exemplul 1:

Nume: Buton "Selecteaza Ruta" - nefunctional
Cale: Inregistrare $\longrightarrow$ Companie $\longrightarrow$ Rute
Descriere: In momentul in care un utilizator nou se inregistreaza si incearca sa selecteze o ruta oferita default pentru a o modifica, butonul nu functioneaza.
Motiv: Ruta default nu se copiaza bine $\longrightarrow$ Probleme in metoda apelata prin apasarea butonului

Solutie: Crearea unui buton ”Adauga Ruta” astfel incat utilizatorul nou nu mai primeste nimic default (inlaturam problema) si poate sa adauge singur intrarea dorita (oferim alternativa).

Bug report - exemplul 2:

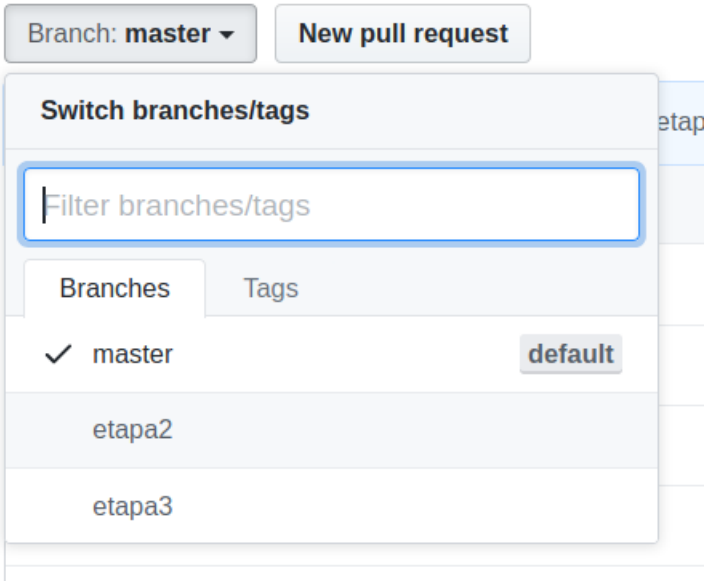
Nume: Buton ”Selecteaza Ruta” - rezultat neasteptat
Cale: Inregistrare $\longrightarrow$ Companie $\longrightarrow$ Rute
Descriere: Pentru 2 utilizatori distincti trebuie sa apara aceleasi curse disponibile (aceasta este singura resursa comuna din baza de date). In momentul in care un utilizator selecteaza o cursa, ea trebuie sa dispara din tabelul general si sa nu mai fie disponibila pentru ceilalti

utilizatori. Problema consta in faptul ca aplicatia permite selectarea multipla (de catre mai multi utilizatori) a unei singure curse
Motiv: La autentificarea unui nou utilizator se creaza cumva copii ale curselor, astfel ca degeaba stergem data din tabel dupa ce a fost selectata deoarece continua sa existe copia ei.
Solutie: Revizuirea pas cu pas a codului. Globalizarea datelor comune astfel incat doar ne referim la ele, nu efectuam copiere.

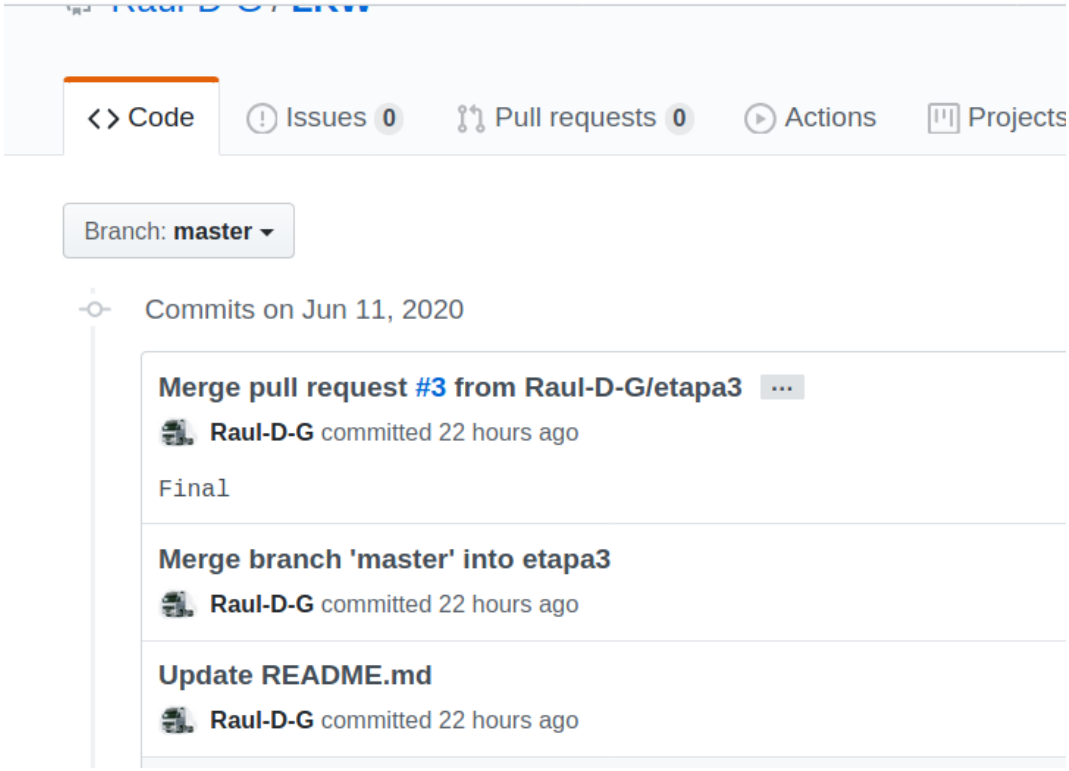
Secretul succesului aplicatiei noastre a constat in *maleabilitate*.Dupa cum puteti vedea si in aceste exemple, ne am mulat in functie de situatiile aparute, am adaugat noi butoane modificand grafica planificata initial, am integrat modificarile facute si nu ne am dat batuti! Atata timp cat ne am putut adapta la evenimentele de care ne am lovit, totul a fost bine!

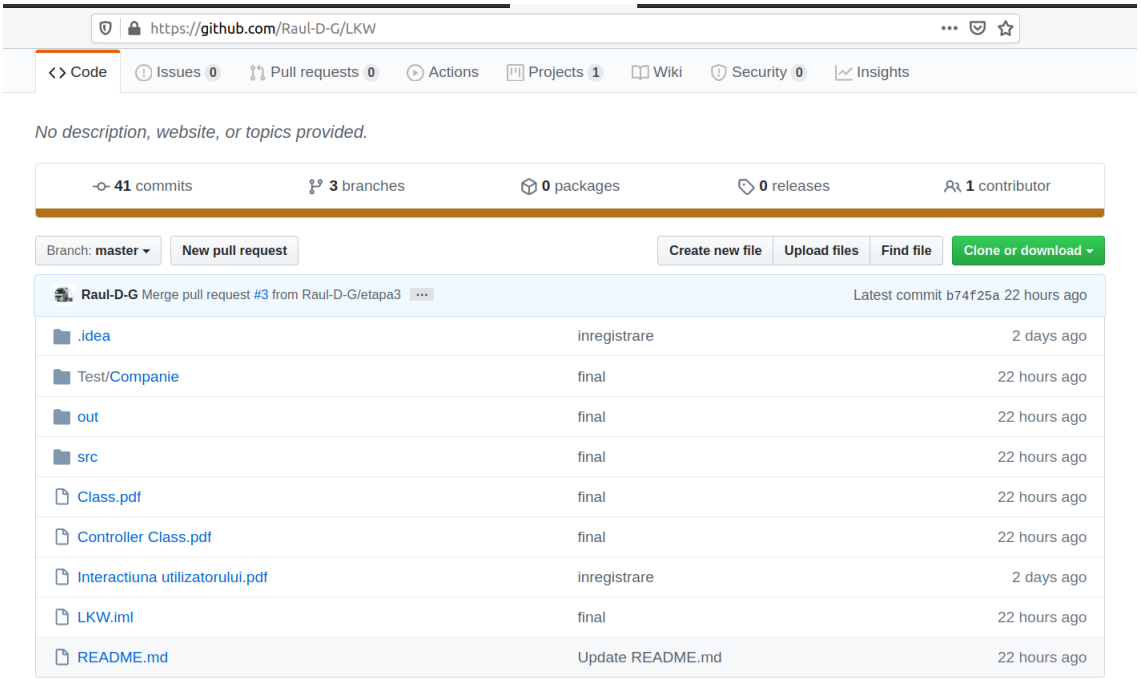
8 Source control

Un proiect mare va avea desigur numeroase etape in productie. Pentru aplicatia noastra *Companie de transport*, am plecat de la 0 cu o simpla idee inspirata din viata reala, care a ajuns sa se materializeze in ceva complet functional si cu un design minimalist si elegant. Inca de la inceput am folosit Github, astfel ca avem imortalizate multe dintre stadiile de dezvoltare.



A fost o munca continua, sustinuta, astfel ca si numarul commit urilor a fost evident, destul de mare. Norocul programatorilor din ziua de astazi este ca exista astfel de "unelte" user friendly precum Github ul, care iti ofera solutii la probleme pe care poate nici nu ai fi stiut ca le ai. O organizare meticuloasa ofera o estetica ingrijita unui proiect, astfel ne am asigurat inca de la inceput ca o sa construim caramida peste caramida si nu va fi totul pus la intamplare.


Code
Issues 0
Pull requests 0
Actions
Projects
Branch: master
Commits on Jun 11, 2020
Merge pull request #3 from Raul-D-G/etapa3
Raul-D-G committed 22 hours ago
Final
Merge branch 'master' into etapa3
Raul-D-G committed 22 hours ago
Update README.md
Raul-D-G committed 22 hours ago

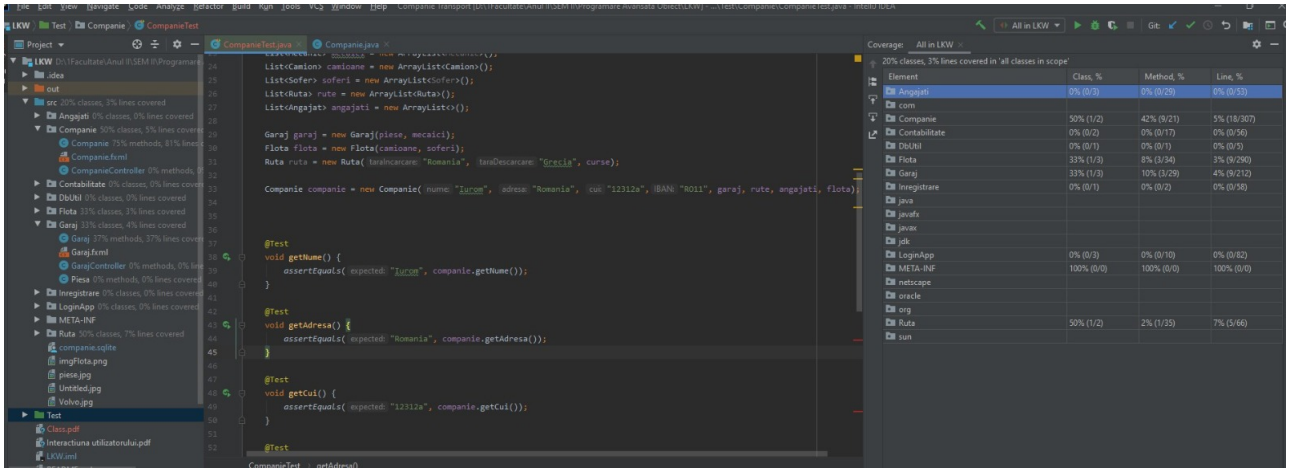

https://github.com/Raul-D-G/LKW
Code
Issues 0
Pull requests 0
Actions
Projects 1
Wiki
Security 0
Insights
No description, website, or topics provided.
41 commits
3 branches
0 packages
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
Raul-D-G Merge pull request #3 from Raul-D-G/etapa3
Latest commit b74f25a 22 hours ago
.idea inregistrare 2 days ago
Test/Companie final 22 hours ago
out final 22 hours ago
src final 22 hours ago
Class.pdf final 22 hours ago
Controller Class.pdf final 22 hours ago
Interactiuna utilizatorului.pdf inregistrare 2 days ago
LKW.iml final 22 hours ago
README.md Update README.md 22 hours ago

9 Teste automate

Pentru a verifica functionalitatea aplicatiei am apelat la testare automata folosind JUnit (versiunea curenta Junit5), un tool simplu dar foarte popular. Acesta a adus completari testelor manuale efectuate de noi si de asemnea este foarte usor de inteles si folosit, inca de la prima vedere.
Am creat teste automate pentru toate metodele "Get", astfel incat sa fim siguri ca datele pe care aplicatia le extrage prin acele metode sunt chiar datele care ne asteptam sa fie extrase.

```
@Test
void getCui() {
    assertEquals("12312a", companie.getCui());
}

@Test
void getIBAN() {
    assertEquals("RO11", companie.getIBAN());
}

@Test
void getGaraj() {
    assertEquals(garaj, companie.getGaraj());
}

@Test
void getRute() {
    assertEquals(rute, companie.getRute());
}

@Test
void getAngajati() {
    assertEquals(angajati, companie.getAngajati());
}

@Test
void getFlota() {
    assertEquals(flota, companie.getFlota());
}
```

La rulare, acestea au demonstrat ca supozitiile noastre au fost corecte. Vom arata un moment cand a fost efectuata optiunea programului de "Run with Coverage ", care ne arata, cat de fapt, a fost testat din codul noastru.



Testarea unitara s a impus in ultima perioada in dezvoltarea proiector scrise in limbajul Java, pe masura aparitiei acestor utilitare gratuite de testare a claselor (precum JUnit) care au contribuit la cresterea vitezei de programare si micsorarea drastica a numarului de "bug uri". Astfel, clasele de test au marit increderea noastra in proiect si ne au permis sa urmarim mai usor unele din cerintele initiale de implementare a proiectului.

10 Incheiere

In incheiere va invitam sa urmariti codul efectiv, disponil la adresa: `https://github.com/Raul-D-G/LKW`.
Sfatul nostru este sa clonati repository -ul, si sa testati personal aplicatia pentru a decide ce parere aveti cu adevarat despre acest concept. Suntem mandrii de rezultatul nostru si speram ca intr o buna zi prezentul proiect va ajunge la nivelul la care acum poate doar spera..