Universitatea Tehnică "Gheorghe Asachi" din Iași
Facultatea de Automatică și Calculatoare
Specializarea Calculatoare și Tehnologia Informației

Baze de date

Aplicație pentru gestionarea clienților, contoarelor și facturilor într-o companie ce furnizeaza energie electrică

Coordonator,
Cristian Butincu

Student,
Radu-Andrei Budeanu
Grupa: 1309A

Iași, 2020

Descrierea proiectului

Aceasta aplicatie are drept scop gestionarea unei baze de date folosita de o companie ce furnizeaza energie electrică, pentru a ușura prelucrarea datelor referitoare la clienți firmei și la facturile care sunt generate pentru fiecare contract.

Aplicația a fost gandita sa aibă ca țintă 3 categorii de utilizatori.

ADMIN-ul este cel care are control total asupra tabelelor importante din baza de date. Acesta poate sa insereze, să șteargă sau să actualizeze cele mai importante entități din baza de date, precum clienți, contracte, tipuri de servicii, angajați. De asemenea, acesta poate sa genereze facturile și poată sa vadă ce verificari au făcut angajații responsabili cu citirea contoarelor.

ANGAJATUL este persoana care are rolul de a verifica contoarele din fiecare gospodarie și de a actualiza indexul real al contoarelor în baza de date. Acesta poate sa instaleze sau sa dezinstaleze un contor dintr-o gospodarie, folosind aplicația pentru a salva datele respective sau pentru a le șterge.

UTILIZATORUL este reprezentat de către clientul ce beneficiază de serviciile companiei. Acesta poate folosi aplicația pentru a-și verifica facturile și pentru a putea trimite indexul curent către companie pentru a putea fi prelucrat.

Fiecare tip de utilizator trebuie mai intai sa se conecteze pentru a putea folosi aplicația prin intermediul unui ID și a unei parole. În cazul ADMINILOR, conturile se vor crea de către administratorul bazei de date, acesta primind un ID unic și o parola. În cazul ANGAJATILOR sau a CLIENȚILOR, aceștia vor trebui mai intai sa își creeze un cont. În cazul unui client, acesta trebuie sa foloseasca codul unic de client pe care îl primește atunci cand încheie un contract cu firma respectivă. În cazul angajatului, acesta va primi codul său unic pentru a putea sa își creeze contul.

Tehnologii folosite

NetBeans

Pentru realizarea aplicației, s-a folosit un tool foarte cunoscut pentru realizarea interfeței grafice(GUI) și anume **NetBeans**.

NetBeans este un software dezvoltat de către *Apache Software Foundation* ce permite crearea de aplicații(preponderent în Java, dar nu numai) folosind diverse componente modulare.

Java

Limbajul de programare folosit este **Java** datorită suportului OOP, dar și a compatibilitatii sale cu orice sistem de operare datorită JVM-ului(Java Virtual Machine).

SWING

Swing este un GUI lightweight utilizat pentru a crea diverse aplicații independente de platforma. Componentele Swing sunt scrise in Java, facand parte din Java Foundation Classes(JFC).

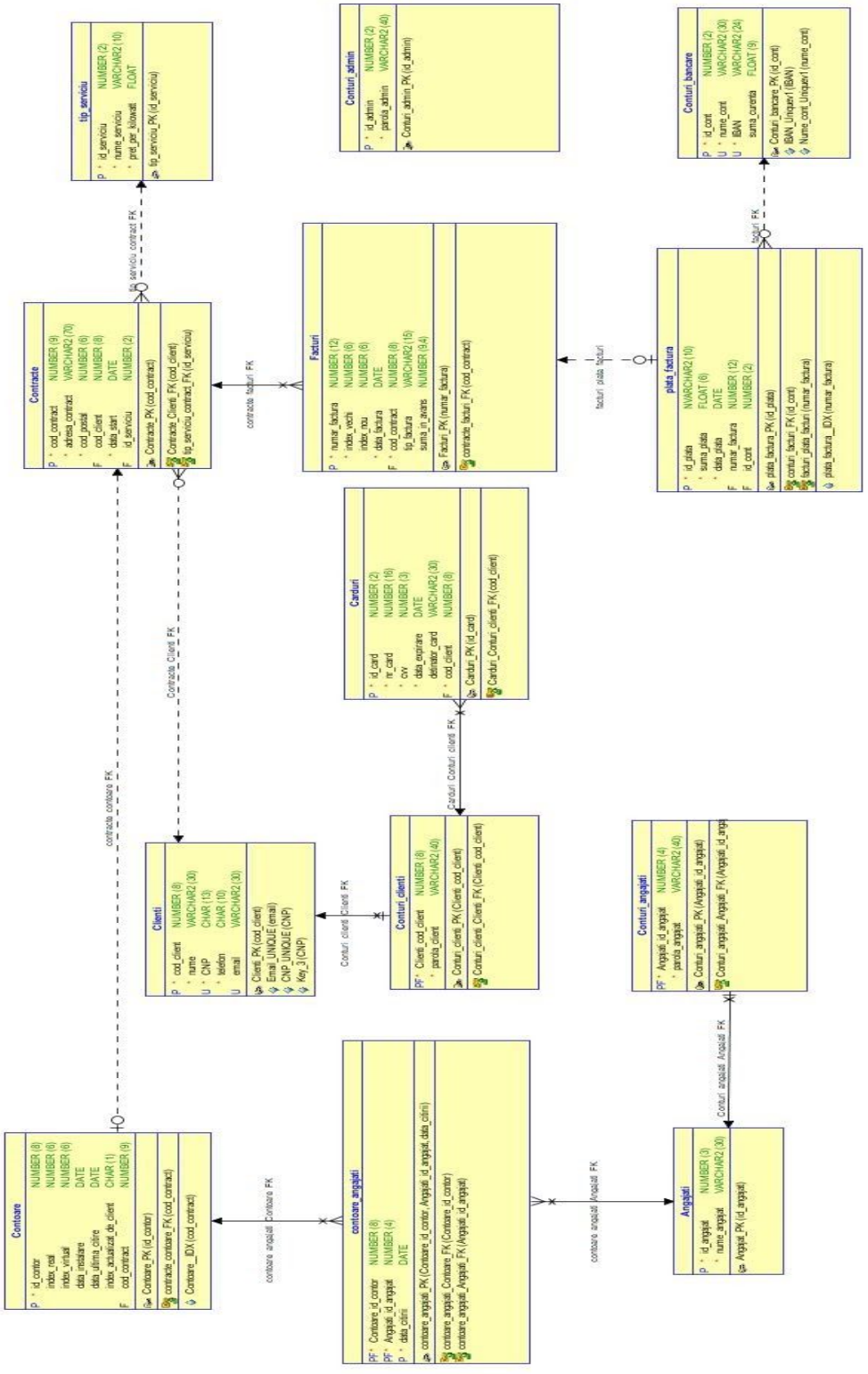
AWT

AWT vine de la Abstract Windows Toolkit. Acesta este un API(Application Programming Interface) pentru dezvoltarea de GUI sau de aplicații bazate pe ferestre Java. A fost dezvoltat de către Sun Microsystems in 1995. Este foarte utilizat deoarece este generat de către sistemul de operare gazda. Conține o mulțime de clase și metode, pentru a crea interfețe grafice și pentru a le gestiona.

Oracle JDBC Driver

Acest driver a fost folosit pentru a se putea realiza o conexiune către o baza de date Oracle 11g cu ajutorul limbajului Java.

Structura și inter relationarea tabelelor



Iași, 2020

Pentru a realiza implementarea acestei baze de date, s-au folosit 3 tipuri de relații: 1:n, 1:1 și n:n.

Între tabelele **Clienți** și **Contracte** s-a definit o relație **1:n,** deoarece un singur client poate avea încheiate mai multe contracte cu respectivă firma(de exemplu, dacă are mai multe locuințe). În mod invers, un singur contract aparține unei singure persoane, acordul fiind încheiat între firma și o singura persoană. Legatura dintre cele doua tabele este realizata cu ajutorul atributului **cod_client**, reprezentand codul unic prin care este identificat un client.

Între **Contracte** și **Contoare** exista o relatie **one-to-one**, deoarece, după încheierea unui contract, firma va asigura instalarea gratuită a unui singur contor. Practic, o gospodarie sau o firma care este conectată la rețeaua electrică, va primi doar un singur dispozitiv pentru măsurarea consumului. Legătura dintre acestea se realizeaza cu ajutorul campului **cod_contract**.

Pentru tabela **Contoare** și tabela **Angajați**, relația este una **many-to-many**. Logica pe care s-a mers este aceea ca un singur angajat poate sa verifice mai multe contoare montate de către firma. În sens invers, un contor poate fi verificat de mai mulți angajați. Pentru ca tabela sa se afle în 3FN, aceasta relatie se va sparge in doua, rezultand, cum este normal, două relatii one-to-many. Așadar, conexiunea dintre cele doua tabele se va realiza cu ajutorul unei alte tabele, **contoare_angajati**, care contine cheia primara din fiecare tabela initiala.

Pentru tabelele **Contracte** și **Facturi**, s-a definit o relatie **one-to-many**, deoarece un contract generează mai multe facturi pană la o eventuala incheiere, pe cand o singură factură aparține doar unui contract, și este imposibil sa se genereze pentru mai multe contracte. Legatura dintre cele doua entitati se realizeaza cu ajutorul atributului **cod_contract,** care este prezent în fiecare factură.

În cazul relației dintre tabela **tip_serviciu** și tabela **Contracte**, aceasta este de tipul **one-to-many**. Pentru fiecare contract încheiat, în funcție de persoana care il încheie(persoana fizica sau juridica), trebuie definit un tip de serviciu. Un utilizator casnic va avea un preț diferit per kilowatt fata de un utilizator industrial. Așadar, un tip de serviciu poate să apară în mai multe contracte, pe când un contract nu se încheie decât pentru un singur serviciu. Atributul care leagă cele două tabele este **id_serviciu**, un identificator unic pentru fiecare tip de serviciu.

Relația dintre **Facturi** și **plata_factura** este una de tipul **one-to-one**. Este oarecum logic, deoarece pentru o factura se poate genera o singură plată validă, iar o plată se realizeaza doar pentru o factura, astfel evitandu-se coliziune. Legătura dintre tabele se realizează cu ajutorul atributului **numar_factura**.

Pentru relația dintre **plata_factura** și **Conturi_bancare** s-a definit tipul **n:1.** În mod clar, într-un cont bancar se pot realiza mai multe plăti, dar o singura plata se poate realiza numai și numai într-un singur cont bancar. Relația dintre tabele este data de atributul **id_cont**, care reprezintă specificatorul unic pentru fiecare cont în parte.

Constrângeri

Pentru a asigura că informațiile din baza de date prezintă o corectitudine minimală, s-au definit o serie de constrângeri, menite sa impiedice inserarea unor date hazardate.

Pentru diferitele atribute care nu pot fi valori mai mici sau egale cu 0, s-a introdus o constrangere de tipul nume_atribut > 0, cum este cazul pentru: **index_curent** din tabela **Contoare**, **pret_per_kilowatt**, din tabela **tip_serviciu** sau **suma_plata** din tabela **plata_factura**.

Pentru inserarea unui email în tabela **Clienti**, se verifica dacă acesta este în formatul specific folosind o expresie regulată, utilizand functia **REGEXP_LIKE**. La fel s-a procedat și pentru verificarea numelor clienților sau al angajatilor, neputand sa fie introduse nume care conțin cifre sau altfel de caractere decat litere.

Pentru atributul **data_ultima_citire** din tabela **Contoare** se verifica la fiecare INSERT sau UPDATE sa fie mai mic sau egal cu data curenta, deoarece se presupune ca o citire este validă doar dacă s-a realizat deja. În același timp, la fiecare INSERT sau UPDATE al unei intrări în aceasta tabela, se verifica pentru a nu se putea insera o data de start mai mare decat data ultimei citiri. De asemenea, pentru atributul **data_start** din tabela **Contracte**, s-a definit o constrângere pentru a nu se putea insera o data mai mare decat data curentă. Practic, un contract nu poate fi creat în viitor, ci doar până în momentul inserarii.

În cazul câtorva atribute, s-a definit și o constrangere de tipul UNIQUE pentru a asigura unicitatea. Astfel de constrângere s-a definit pentru un **email**, deoarece nu poate exista un email pentru mai mulți angajați. Aceeași poveste este valabilă și pentru numele unui cont si **IBAN-ul** asociat acestuia, din tabela **Conturi_bancare**.

Pentru a valida numărul de telefon introdus în tabela **Client**, s-a adaugat o constrângere care verifica dimensiunea acestuia(10 cifre) și în același timp dacă numărul are prefixul specific rețelelor de telefonie din România('07xx-xxx-xxx'').

Din cauză că facturile generate pot fi doar de trei tipuri, s-a creat o constrângere care limitează valoarea atributului **tip_factura** la una din variantele: ESTIMARE, CITIRE_CLIENT, CITIRE_ANGAJAT specifice pentru facturile care sunt generate cu un index estimativ, care sunt generate după ce clientul a făcut o actualizare de index sau care au fost generate după ce angajații au realizat cate o verificare a contoarelor.

O ultima constrângere care a fost adăugată este cea pentru atributul **index_virtual** din tabela **Contoare**. La fiecare actualizare se verifică prin intermediul unui trigger dacă valoarea noului index inserat este mai mare decat cea a vechiului index. Astfel, se poate evita ca un client sa ofere informații care ar fi în favoarea lui pentru plata unei facturi.

Conectarea la baza de date

Pentru conectarea la baza de date, după cum s-a precizat mai sus, s-a folosit driverul oficial de la Oracle, și anume **Oracle Database 11.2.0.4 JDBC Driver.**

Pentru a se defini conexiunea în cadrul aplicației, s-a pornit de la interfața DatabaseConnection pentru a defini o conexiune generică la o baza de date. Implementarea concreta este reprezentata de clasa OracleSQLConnection ce incapsuleaza toate datele și metodele necesare pentru a realiza conexiunea la baza de date și interogarea acesteia, respectiv inserarea, ștergerea sau actualizarea informațiilor.

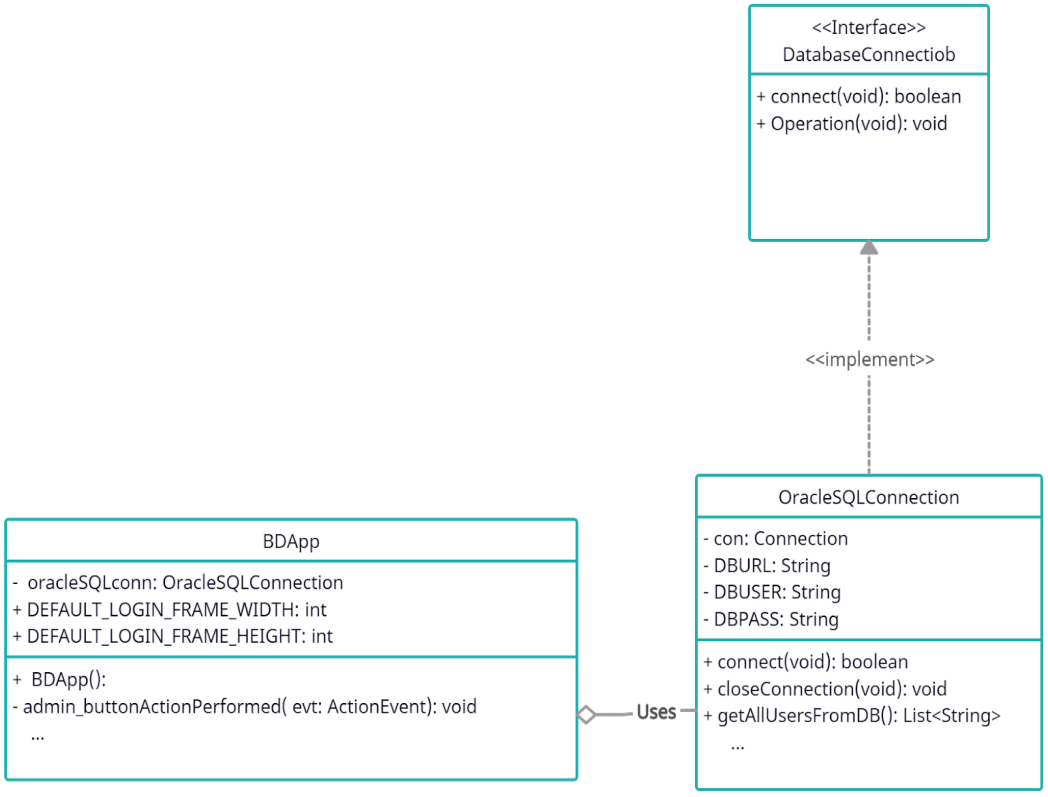


Diagrama principala de clase a aplicației

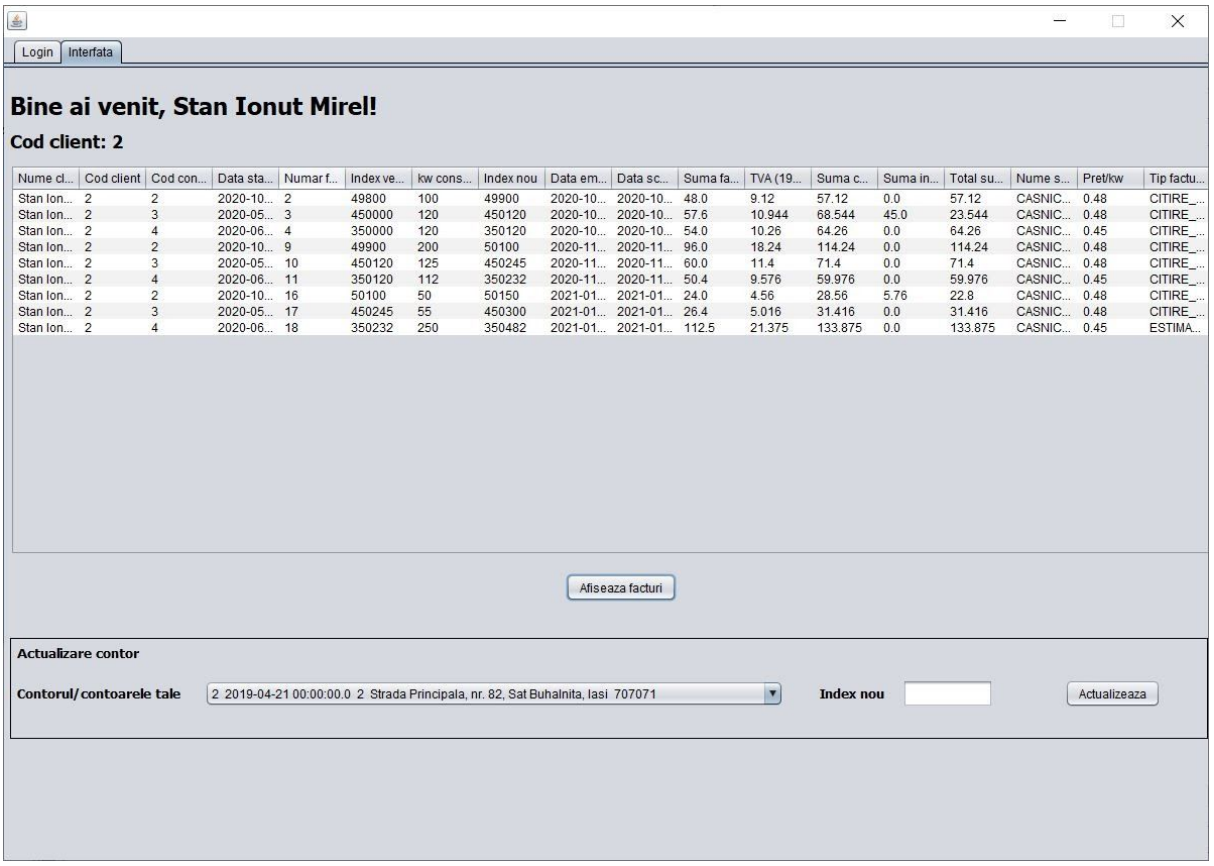
Instrucțiuni SQL

Pentru obținerea informațiilor din baza de date, funcția SELECT s-a folosești cel mai des. De exemplu, pentru afișarea informațiilor de interes pentru toate facturile unui client, s-a folosit metoda de mai jos:

```
public List<String> getAllBilsByUser(long user_id){
 connect();

 List<String> bils = new ArrayList<>();

 try{
   Statement statement = con.createStatement();

   String query = "SELECT clienti.nume, clienti.cod_client,contracte.cod_contract,contracte.data_start, facturi.numar_factura,\n" +
         "facturi.index_vechi,(facturi.index_nou-facturi.index_vechi) as kw_consumati,facturi.index_nou, facturi.data_factura, facturi.data_factura + 14 as data_scadenta,\n" +
         "(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) as suma_fara_TVA,\n" +
         "0.19*(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) as TVA,\n" +
         "(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) + \n" +
         "0.19*(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) as SUMA_CONSUM," +
         "facturi.suma_in_avans,(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) + \n" +
                  "0.19*(facturi.index_nou-facturi.index_vechi)*(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) - facturi.suma_in_avans as
total_de_plata," +
         "(SELECT nume_serviciu FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) as nume_serviciu,\n" +
         "(SELECT pret_per_kilowatt FROM tip_serviciu WHERE contracte.id_serviciu=tip_serviciu.id_serviciu) as pret_per_kw,\n" +
         "facturi.tip_factura\n"+
         "FROM contracte, clienti, facturi WHERE \n" +
         "clienti.cod_client = contracte.cod_client AND contracte.cod_contract = facturi.cod_contract AND clienti.cod_client = ";
```

Iași, 2020

```
    query += user_id;

    query += " ORDER BY numar_factura";
    ResultSet rs = statement.executeQuery(query);

    String new_list_val = "";

    while(rs.next()) {
      new_list_val += rs.getString("nume") + "  ";
      new_list_val += rs.getLong("cod_client") + "  ";
      new_list_val += rs.getLong("cod_contract") + "  ";
      new_list_val += rs.getString("data_start") + "  ";
      new_list_val += rs.getLong("numar_factura") + "  ";
      new_list_val += rs.getLong("index_vechi") + "  ";
      new_list_val += rs.getLong("kw_consumati") + "  ";
      new_list_val += rs.getLong("index_nou") + "  ";
      new_list_val += rs.getString("data_factura") + "  ";
      new_list_val += rs.getString("data_scadenta") + "  ";
      new_list_val += rs.getFloat("suma_fara_tva") + "  ";
      new_list_val += rs.getFloat("tva") + "  ";
      new_list_val += rs.getFloat("suma_consum") + "  ";
      new_list_val += rs.getFloat("suma_in_avans") + "  ";
      new_list_val += rs.getFloat("total_de_plata") + "  ";
      new_list_val += rs.getString("nume_serviciu") + "  ";
      new_list_val += rs.getFloat("pret_per_kw") + "  ";
      new_list_val += rs.getString("tip_factura");

      bils.add(new_list_val);

      new_list_val = "";
    }

    rs.close();
    statement.close();
    closeConnection();
  }catch(SQLException e){
    System.out.println(e.getMessage());
    JOptionPane.showMessageDialog(null,"Eroare necunoscuta la citirea facturilor din baza de date!","Eroare",JOptionPane.WARNING_MESSAGE);

    closeConnection();
  }

  return bils;
 }
```



Nume cl...	Cod client	Cod con...	Data sta...	Numar f...	Index ve...	kw cons...	Index nou	Data em...	Data sc...	Suma fa...	TVA (19...	Suma c...	Suma in...	Total su...	Nume s...	Pret/kw	Tip factu...
Stan Ion...	2	2	2020-10...	2	49800	100	49900	2020-10...	2020-10...	48.0	9.12	57.12	0.0	57.12	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	3	2020-05...	3	450000	120	450120	2020-10...	2020-10...	57.6	10.944	68.544	45.0	23.544	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	4	2020-06...	4	350000	120	350120	2020-10...	2020-10...	54.0	10.26	64.26	0.0	64.26	CASNIC...	0.45	CITIRE_...
Stan Ion...	2	2	2020-10...	9	49900	200	50100	2020-11...	2020-11...	96.0	18.24	114.24	0.0	114.24	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	3	2020-05...	10	450120	125	450245	2020-11...	2020-11...	60.0	11.4	71.4	0.0	71.4	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	4	2020-06...	11	350120	112	350232	2020-11...	2020-11...	50.4	9.576	59.976	0.0	59.976	CASNIC...	0.45	CITIRE_...
Stan Ion...	2	2	2020-10...	16	50100	50	50150	2021-01...	2021-01...	24.0	4.56	28.56	5.76	22.8	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	3	2020-05...	17	450245	55	450300	2021-01...	2021-01...	26.4	5.016	31.416	0.0	31.416	CASNIC...	0.48	CITIRE_...
Stan Ion...	2	4	2020-06...	18	350232	250	350482	2021-01...	2021-01...	112.5	21.375	133.875	0.0	133.875	CASNIC...	0.45	ESTIMA...

Interfata grafica din panoul utilizatorului

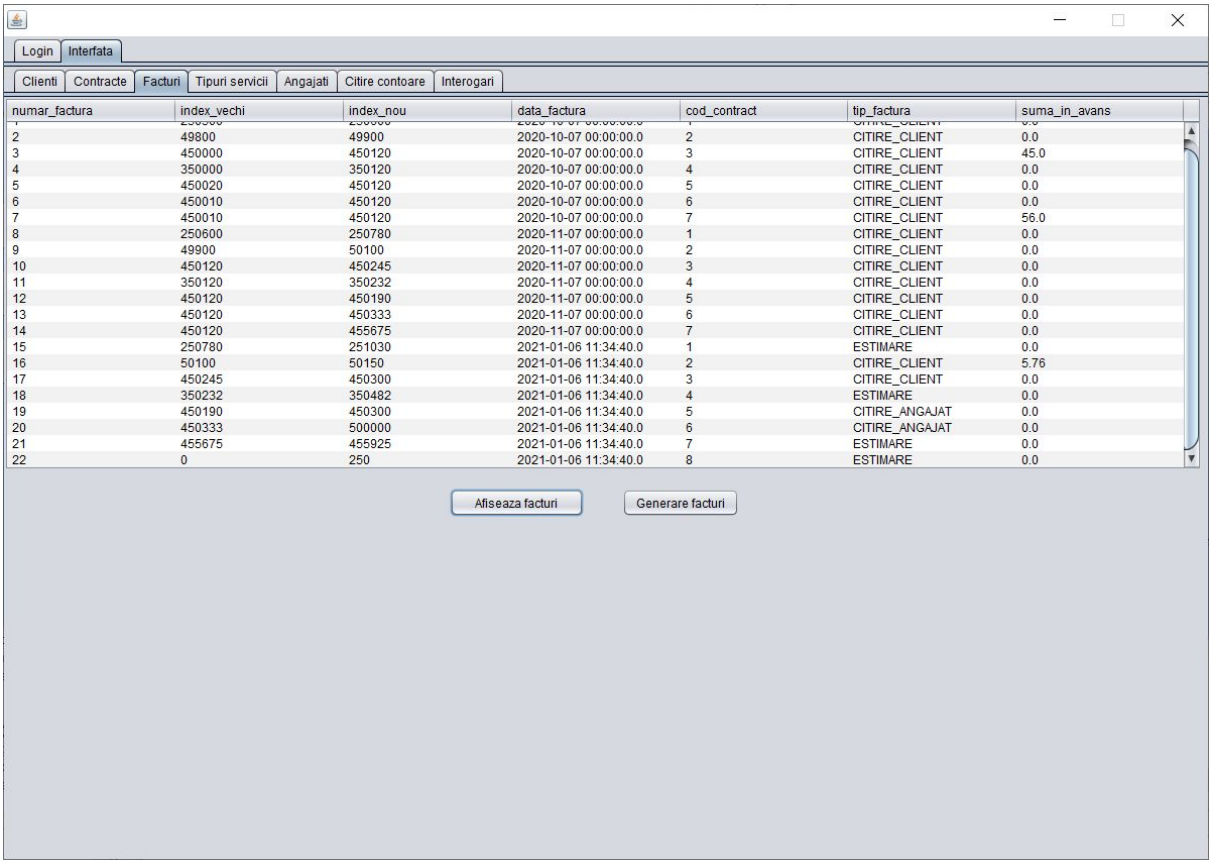
De asemenea, pentru actualizarea informațiilor s-au folosit instrucțiunea UPDATE iar pentru ștergerea acestora sau inserarea acestora, s-a folosit instrucțiunea DELETE, respectiv INSERT.

```
public void insertNewContract( String contract_address, int postal_code, long client_code, String start_date, int service_id) throws ParseException{
  connect();

  try{
    Statement statement = con.createStatement();

    String query = "INSERT INTO Contracte (adresa_contract,cod_postal,cod_client,data_start,id_serviciu) VALUES ('";
    query +=  contract_address + "'," + postal_code + "," + client_code + ",";
    if(start_date != null)
    {
      query += "TO_DATE('" + start_date + "', 'YYYY-MM-DD')," + service_id + ")";
    }
    else
    {
      JOptionPane.showMessageDialog(null,"Data start invalida la inserarea unui nou contract!","Eroare",JOptionPane.WARNING_MESSAGE);
    }

    System.out.println(query);


    statement.executeUpdate(query);

    statement.close();
    closeConnection();
  }catch(SQLException e){
    System.out.println(e.getMessage());
    JOptionPane.showMessageDialog(null,"Eroare necunoscuta la inserarea unui client in baza de date!","Eroare",JOptionPane.WARNING_MESSAGE);

    closeConnection();
```

Iași, 2020

```
    }
  }
public void deleteContract(long contract_id){
    connect();
    try{
      Statement statement = con.createStatement();

      String query = "DELETE FROM Contracte WHERE cod_contract=" + String.valueOf(contract_id);

      statement.executeUpdate(query);

      statement.close();
      closeConnection();
    }catch(SQLException e){
      System.out.println(e.getMessage());
      JOptionPane.showMessageDialog(null,"Eroare necunoscuta la stergerea unui contract din baza de date!","Eroare",JOptionPane.WARNING_MESSAGE);

      closeConnection();
    }
  }
```

Pentru generarea unor noi facturi, s-a folosit o tranzactie sub forma unei proceduri SQL. Aceasta procedura are rolul de a verifica pentru fiecare contract dacă indexul a fost citit de către angajat sau client. În caz ca a fost citit de niciunul dintre cei doi, atunci factura se va genera estimativ. Tranzacția ține cont și de plățile anterioare, astfel incat dacă un client a plătit mai mult pentru o factura, suma restantă va fi scăzută din facturile viitoare.



numar_factura	index_vechi	index_nou	data_factura	cod_contract	tip_factura	suma_in_avans
2	49800	49900	2020-10-07 00:00:00.0	2	CITIRE_CLIENT	0.0
3	450000	450120	2020-10-07 00:00:00.0	3	CITIRE_CLIENT	45.0
4	350000	350120	2020-10-07 00:00:00.0	4	CITIRE_CLIENT	0.0
5	450020	450120	2020-10-07 00:00:00.0	5	CITIRE_CLIENT	0.0
6	450010	450120	2020-10-07 00:00:00.0	6	CITIRE_CLIENT	0.0
7	450010	450120	2020-10-07 00:00:00.0	7	CITIRE_CLIENT	56.0
8	250600	250780	2020-11-07 00:00:00.0	1	CITIRE_CLIENT	0.0
9	49900	50100	2020-11-07 00:00:00.0	2	CITIRE_CLIENT	0.0
10	450120	450245	2020-11-07 00:00:00.0	3	CITIRE_CLIENT	0.0
11	350120	350232	2020-11-07 00:00:00.0	4	CITIRE_CLIENT	0.0
12	450120	450190	2020-11-07 00:00:00.0	5	CITIRE_CLIENT	0.0
13	450120	450333	2020-11-07 00:00:00.0	6	CITIRE_CLIENT	0.0
14	450120	455675	2020-11-07 00:00:00.0	7	CITIRE_CLIENT	0.0
15	250780	251030	2021-01-06 11:34:40.0	1	ESTIMARE	0.0
16	50100	50150	2021-01-06 11:34:40.0	2	CITIRE_CLIENT	5.76
17	450245	450300	2021-01-06 11:34:40.0	3	CITIRE_CLIENT	0.0
18	350232	350482	2021-01-06 11:34:40.0	4	ESTIMARE	0.0
19	450190	450300	2021-01-06 11:34:40.0	5	CITIRE_ANGAJAT	0.0
20	450333	500000	2021-01-06 11:34:40.0	6	CITIRE_ANGAJAT	0.0
21	455675	455925	2021-01-06 11:34:40.0	7	ESTIMARE	0.0
22	0	250	2021-01-06 11:34:40.0	8	ESTIMARE	0.0

Panoul pentru vizualizarea și generarea facturilor

Iași, 2020

Alte doua tranzactii au loc atunci cand un angajat sau un client actualizează indexul unui contor. Pentru fiecare tranzactie s-a definit procedura SQL apelata din Java pentru actualizarea datelor respective din tabelele **Contoare** și **Contoare_anagajati**.

O ultima tranzactie are loc atunci cand un client dorește sa plătească o factură prin intermediul aplicației. Respectiva tranzactie realizeaza o nouă intrare în tabela **plata_factura** și actualizează suma din contul în care se vireaza banii.