DOCUMENTATIE

Tema 3

NUME STUDENT:CIONTE SERGIU IONUT

GRUPA:30228

CUPRINS

1. OBIECTIVUL TEMEI

2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

3. PEOIWCTARE

4. IMPLEMENTARE

5. REZULTATE

6. CONCLUZII

7. BIBLIOGRAFIE

1.OBIECTIVUL TEMEI

Obiectivul principal:

Design-ul si implementarea unei aplicatii care sa modeleze client order-urile pentru un warehouse

Obiective secundare:

Analiza problemei si identificarea cerintelor;

Design-ulaplicatiei de order management;

Implementarea aplicatiei de order management

Testarea aplicatiei de order management;

2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

Cerinte functionale:

Aplicatia trebuie sa permita angajatorului sa adauge un client;

Aplicatia trebuie sa permita angajatorului sa adauge un produs;

Aplicatia trebuie sa permita angajatorului sa creeze o comanda;

Cerinte nonfunctionale:

Aplicatia trebuie sa fie intuitivia si usor de utilizat;

Aplicatia trebuie sa permita angajatorului sa vizualizeze listele de clienti, produse si comenzi;

Scenarii:

Scenariu principal:

Angajatorul selecteaza o optiune sa adauge un produs

Aplicatia va afisa un window pentru a introduce datele produsului;

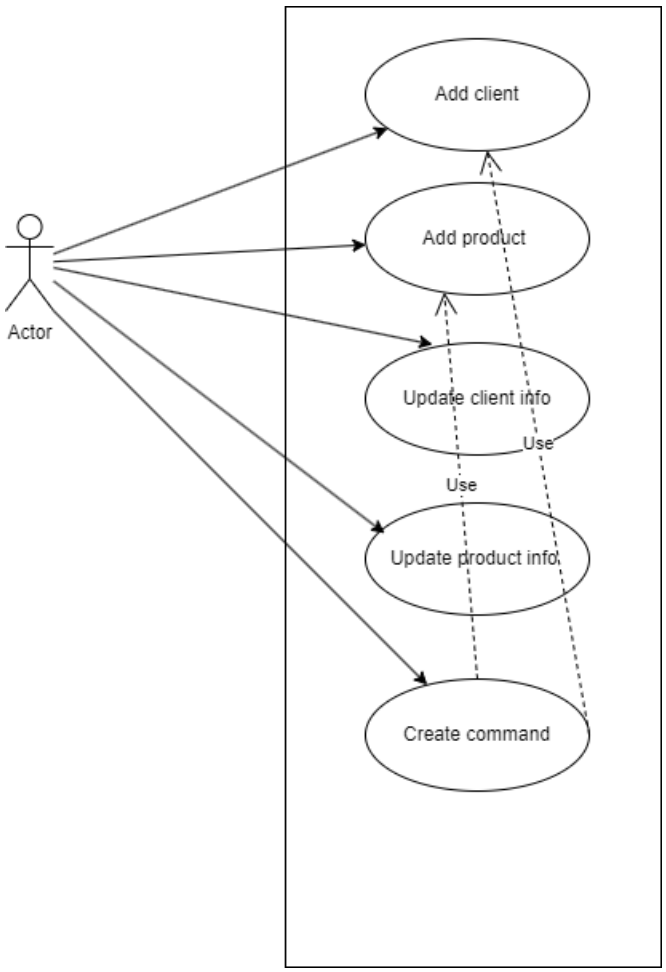
Angajatorul selecteaza butonul insert;

Aplicatia stocheaza produsul in baza de date

Scenariu secundar:

Scenariu secundar:

Angajatorul introduce date gresite afisandu-se un mesaj sau selecteaza o comanda a unui produs introducand o cantitate mai mare decat stocul produsului si in acest caz afisandu-se un mesaj;



3.Proiectare

Diagrama de pachete:

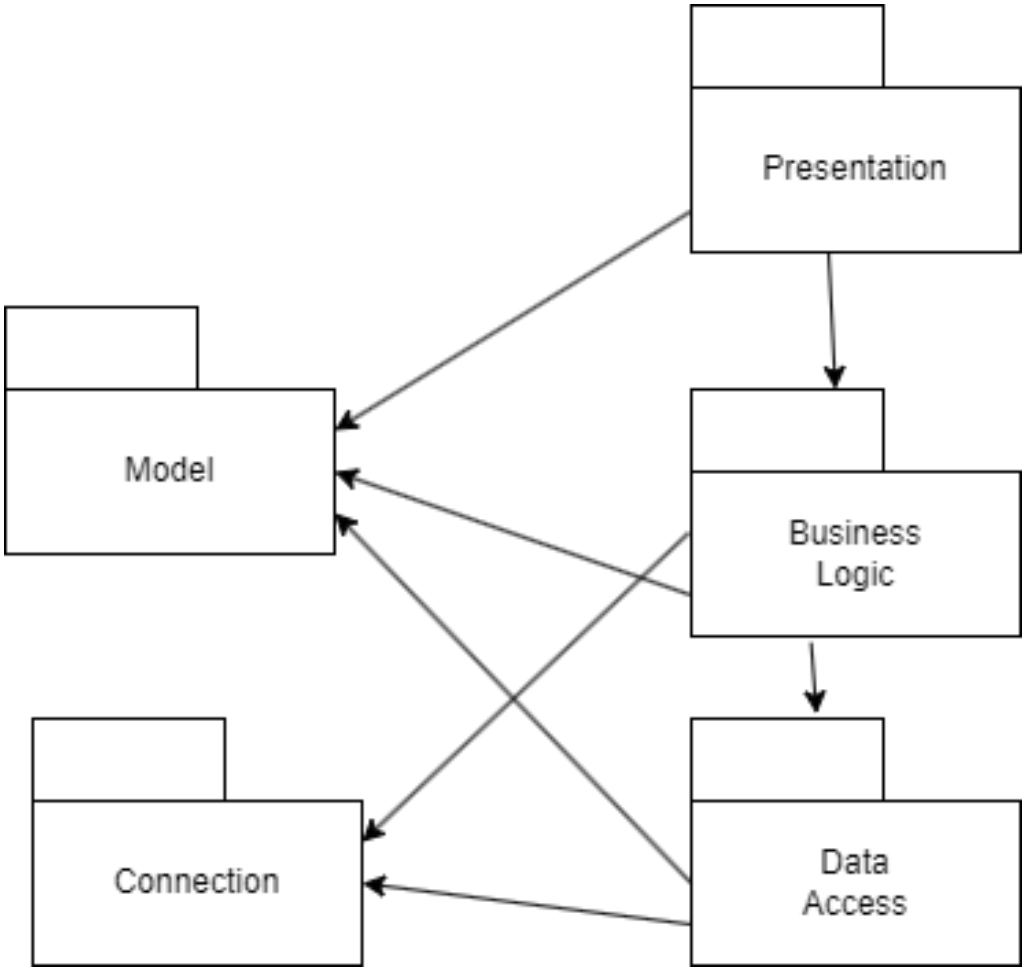
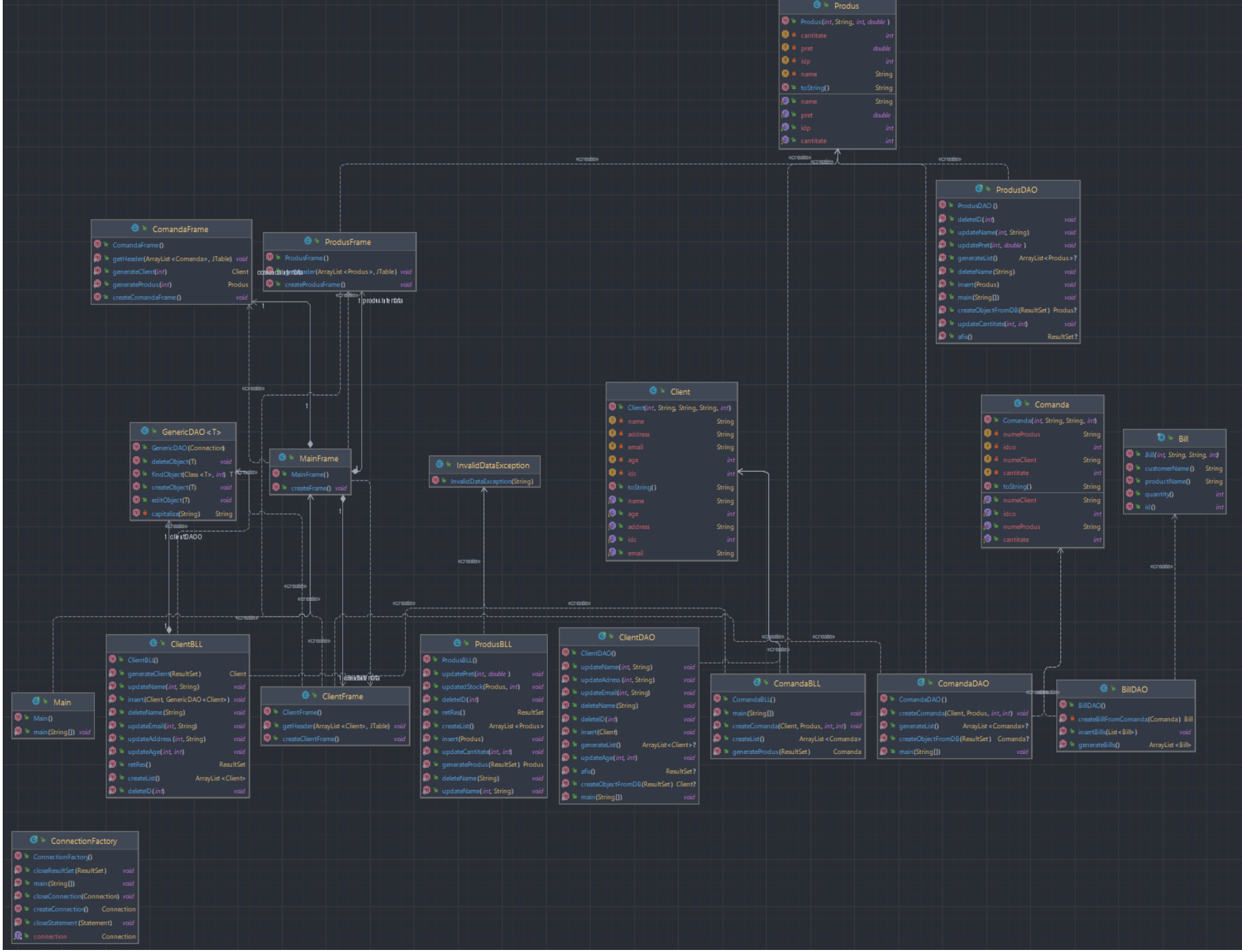


Diagrama de clase

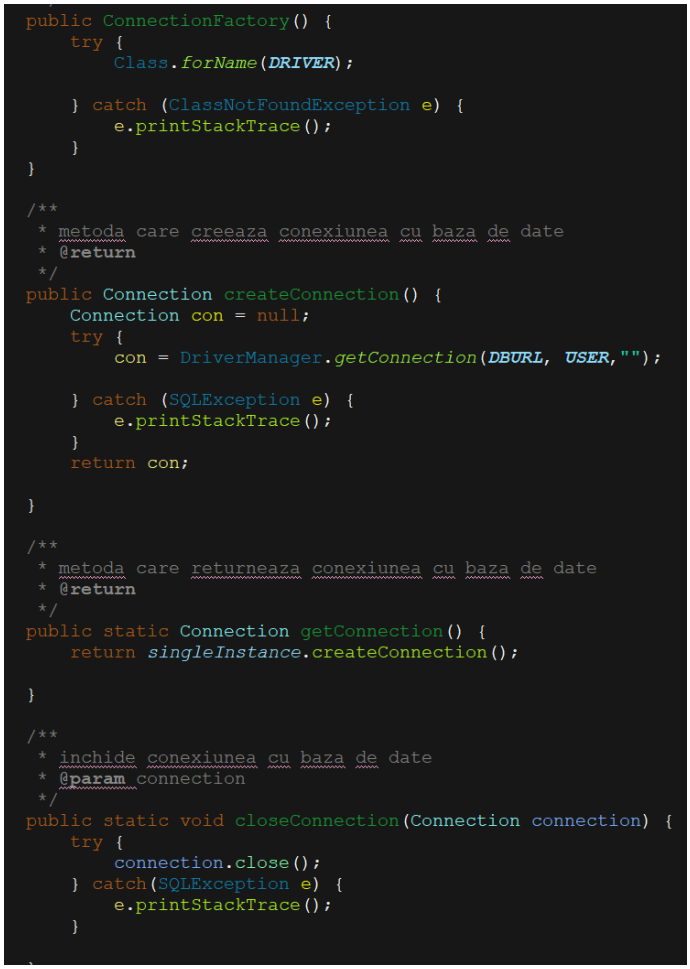

Produs
Produs(int, String, int, double)
cantitate int
pret double
idp int
name String
toString() String
name String
pret double
idp int
cantitate int
«create»
ProdusDAO
ProdusDAO()
deleteID(int) void
updateName(int, String) void
updatePret(int, double) void
generateList() ArrayList<Produs>?
deleteName(String) void
insert(Produs) void
main(String[]) void
createObjectFromDB(ResultSet) Produs?
updateCantitate(int, int) void
afis() ResultSet?
ComandaFrame
ComandaFrame()
getHeader(ArrayList<Comanda>, JTable) void
generateClient(int) Client
generateProdus(int) Produs
createComandaFrame() void
ProdusFrame
ProdusFrame()
createProdusFrame() void
1 produsTerminal
GenericDAO<T>
GenericDAO(Connection)
deleteObject(T) void
findObject(Class<T>, int) T
createObject(T) void
editObject(T) void
capitalize(String) String
1 clientDAO
MainFrame
MainFrame()
createFrame() void
InvalidDataException
InvalidDataException(String)
Client
Client(int, String, String, String, int)
name String
address String
email String
age int
idc int
toString() String
name String
age int
address String
idc int
email String
Comanda
Comanda(int, String, String, int)
numeProdus String
idco int
numeClient String
cantitate int
toString() String
numeClient String
idco int
numeProdus String
cantitate int
Bill
Bill(int, String, String, int)
customerName() String
productName() String
quantity() int
id() int
Main
Main()
main(String[]) void
ClientBLL
ClientBLL()
generateClient(ResultSet) Client
updateName(int, String) void
insert(Client, GenericDAO<Client>) void
deleteName(String) void
updateEmail(int, String) void
updateAddress(int, String) void
updateAge(int, int) void
retRes() ResultSet
createList() ArrayList<Client>
deleteID(int) void
ClientFrame
ClientFrame()
getHeader(ArrayList<Client>, JTable) void
createClientFrame() void
1 clientTerminal
ProdusBLL
ProdusBLL()
updatePret(int, double) void
updatedStock(Produs, int) void
deleteID(int) void
retRes() ResultSet
createList() ArrayList<Produs>
insert(Produs) void
updateCantitate(int, int) void
generateProdus(ResultSet) Produs
deleteName(String) void
updateName(int, String) void
ClientDAO
ClientDAO()
updateName(int, String) void
updateAdress(int, String) void
updateEmail(int, String) void
deleteName(String) void
deleteID(int) void
insert(Client) void
generateList() ArrayList<Client>?
updateAge(int, int) void
afis() ResultSet?
createObjectFromDB(ResultSet) Client?
main(String[]) void
ComandaBLL
ComandaBLL()
main(String[]) void
createComanda(Client, Produs, int, int) void
createList() ArrayList<Comanda>
generateProdus(ResultSet) Comanda
ComandaDAO
ComandaDAO()
createComanda(Client, Produs, int, int) void
generateList() ArrayList<Comanda>?
createObjectFromDB(ResultSet) Comanda?
main(String[]) void
BillDAO
BillDAO()
createBillFromComanda(Comanda) Bill
insertBills(List<Bill>) void
generateBills() ArrayList<Bill>
ConnectionFactory
ConnectionFactory()
closeResultSet(ResultSet) void
main(String[]) void
closeConnection(Connection) void
createConnection() Connection
closeStatement(Statement) void
connection Connection

4.IMPLEMENTARE

Clasele din model(Client,Produs,Comanda si Bill) contin atributele pentru aceste obiecte precum si gettere si settere.

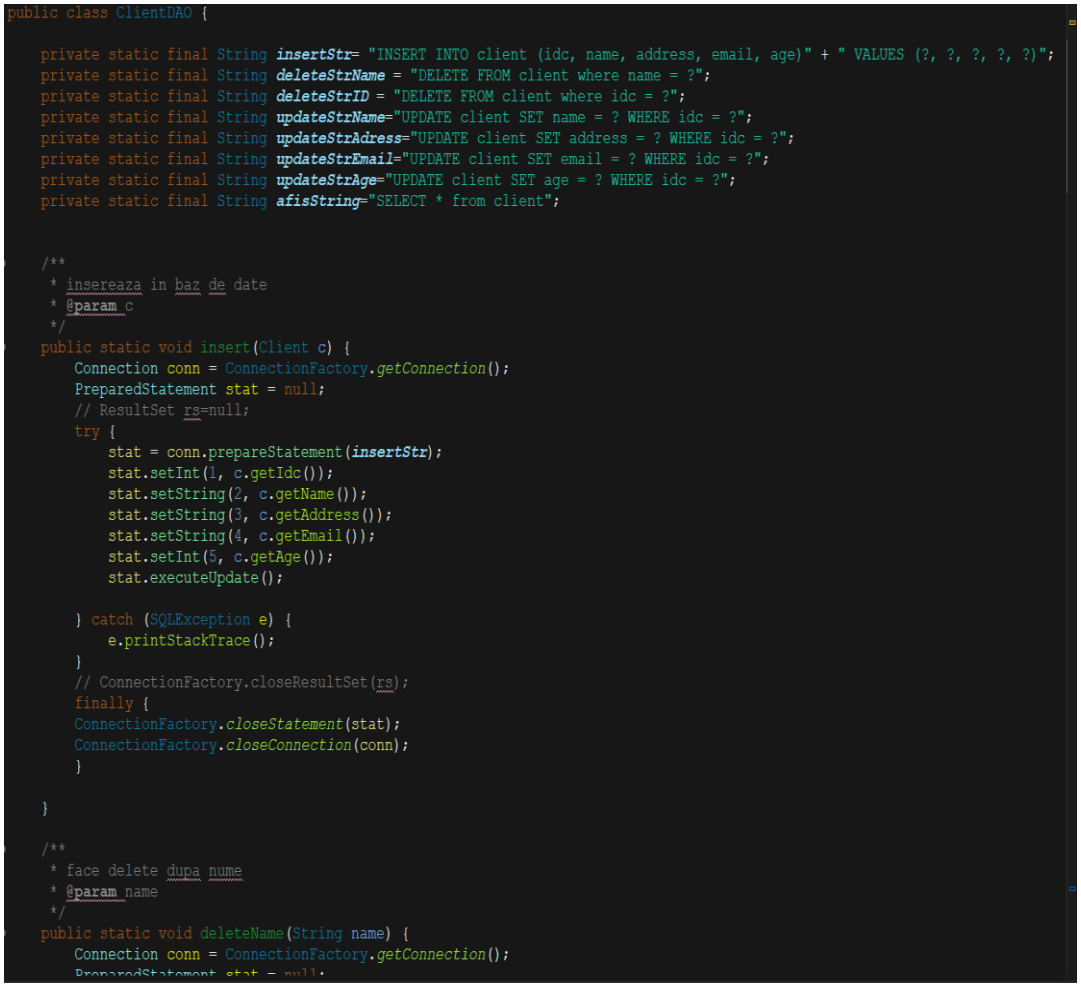
```java
public class Client {
    private int idc;
    private String name;
    private String address;
    private String email;
    private int age;
    /**
     * constructor pentru client
     * @param idc
     * @param name
     * @param address
     * @param email
     * @param age
     */
    public Client(int idc, String name, String address, String email, int age) {
        super();
        this.idc = idc;
        this.name = name;
        this.address = address;
        this.email = email;
        this.age = age;
    }
    /**
     * getter
     * @return
     */
    public int getIdc() {
        return idc;
    }
    /**
     * setter
     * @param idc
     */
    public void setIdc(int idc) {
        this.idc = idc;
    }
    /**
     * getter
     * @return
     */
```

Clasa ConnectionFactory stabileste conexiunea cu baza de date si contine si metoda pentru returnarea conexiunii cu baza de date respectiv pentru inchiderea conexiunii cu baza de date.



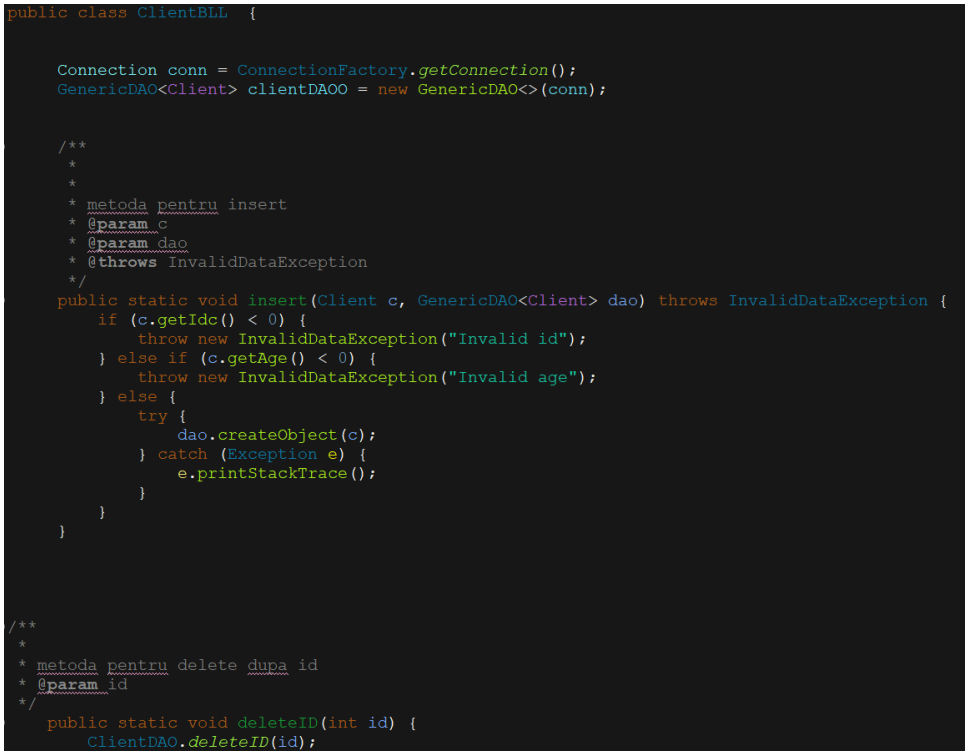
```
public ConnectionFactory() {
    try {
        Class.forName(DRIVER);

    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}

/**
 * metoda care creeaza conexiunea cu baza de date
 * @return
 */
public Connection createConnection() {
    Connection con = null;
    try {
        con = DriverManager.getConnection(DBURL, USER,"");

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return con;

}

/**
 * metoda care returneaza conexiunea cu baza de date
 * @return
 */
public static Connection getConnection() {
    return singleInstance.createConnection();

}

/**
 * inchide conexiunea cu baza de date
 * @param connection
 */
public static void closeConnection(Connection connection) {
    try {
        connection.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
```

Clasele din data access acceseaza baza de date si contin metode pentru insert delete si update pentru client si produs, in timp ce clasa ComandaDAO creeaza o comanda utilizand un client un produs si o anumita cantitate.



```java
public class ClientDAO {

    private static final String insertStr= "INSERT INTO client (idc, name, address, email, age)" + " VALUES (?, ?, ?, ?, ?)";
    private static final String deleteStrName = "DELETE FROM client where name = ?";
    private static final String deleteStrID = "DELETE FROM client where idc = ?";
    private static final String updateStrName="UPDATE client SET name = ? WHERE idc = ?";
    private static final String updateStrAdress="UPDATE client SET address = ? WHERE idc = ?";
    private static final String updateStrEmail="UPDATE client SET email = ? WHERE idc = ?";
    private static final String updateStrAge="UPDATE client SET age = ? WHERE idc = ?";
    private static final String afisString="SELECT * from client";


    /**
     * insereaza in baz de date
     * @param c
     */
    public static void insert(Client c) {
        Connection conn = ConnectionFactory.getConnection();
        PreparedStatement stat = null;
        // ResultSet rs=null;
        try {
            stat = conn.prepareStatement(insertStr);
            stat.setInt(1, c.getIdc());
            stat.setString(2, c.getName());
            stat.setString(3, c.getAddress());
            stat.setString(4, c.getEmail());
            stat.setInt(5, c.getAge());
            stat.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
        // ConnectionFactory.closeResultSet(rs);
        finally {
        ConnectionFactory.closeStatement(stat);
        ConnectionFactory.closeConnection(conn);
        }

    }

    /**
     * face delete dupa nume
     * @param name
     */
    public static void deleteName(String name) {
        Connection conn = ConnectionFactory.getConnection();
        PreparedStatement stat = null;
```

Clasele din business logic implementeaza logica aplicatiei

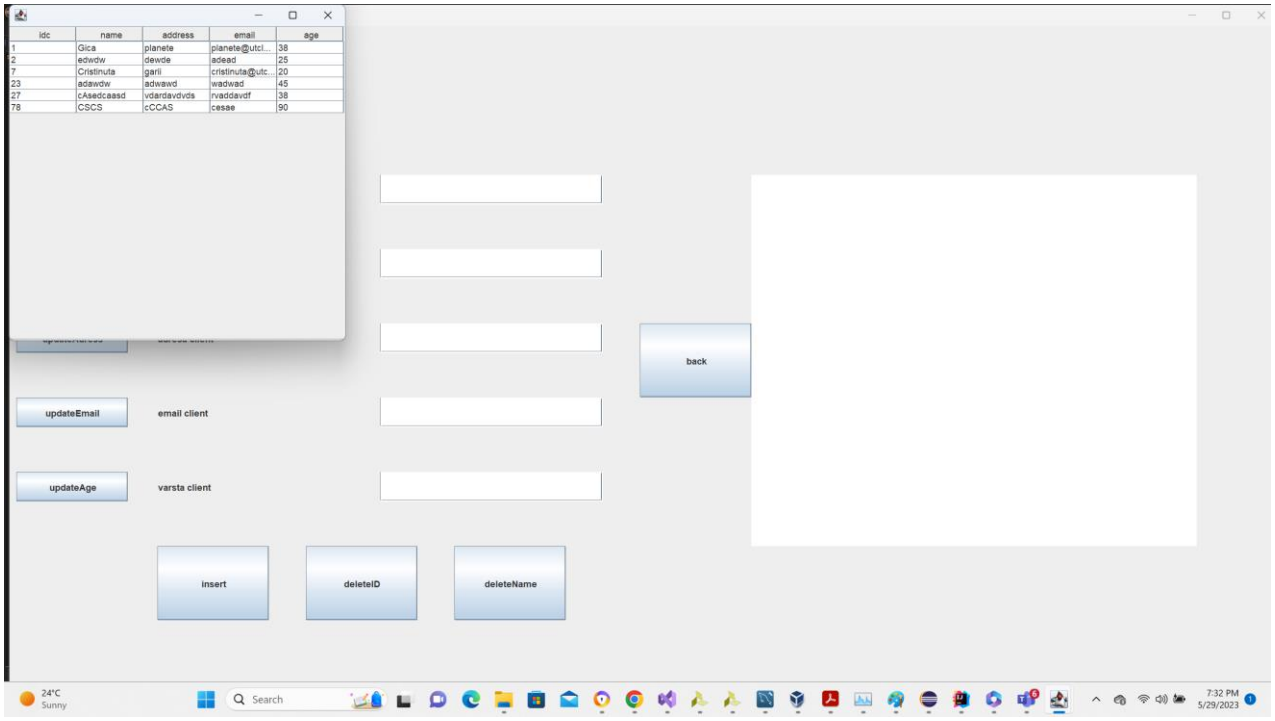


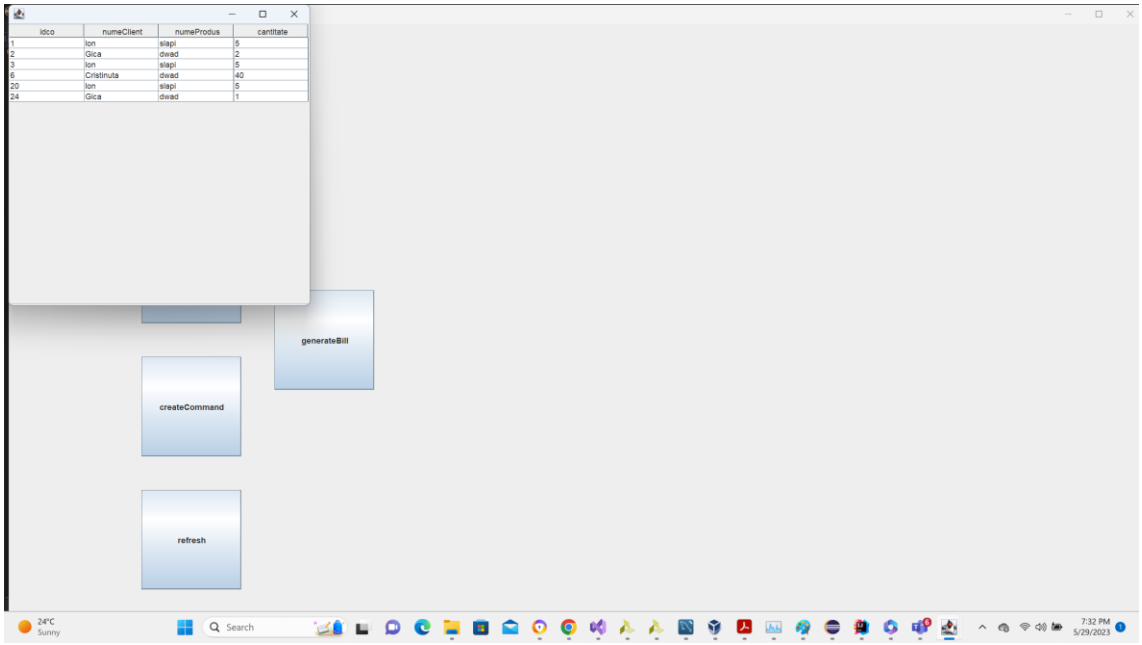
```
public class ClientBLL  {


    Connection conn = ConnectionFactory.getConnection();
    GenericDAO<Client> clientDAOO = new GenericDAO<>(conn);


    /**
     *
     *
     * metoda pentru insert
     * @param c
     * @param dao
     * @throws InvalidDataException
     */
    public static void insert(Client c, GenericDAO<Client> dao) throws InvalidDataException {
        if (c.getIdc() < 0) {
            throw new InvalidDataException("Invalid id");
        } else if (c.getAge() < 0) {
            throw new InvalidDataException("Invalid age");
        } else {
            try {
                dao.createObject(c);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }




/**
 *
 * metoda pentru delete dupa id
 * @param id
 */
   public static void deleteID(int id) {
       ClientDAO.deleteID(id);
```

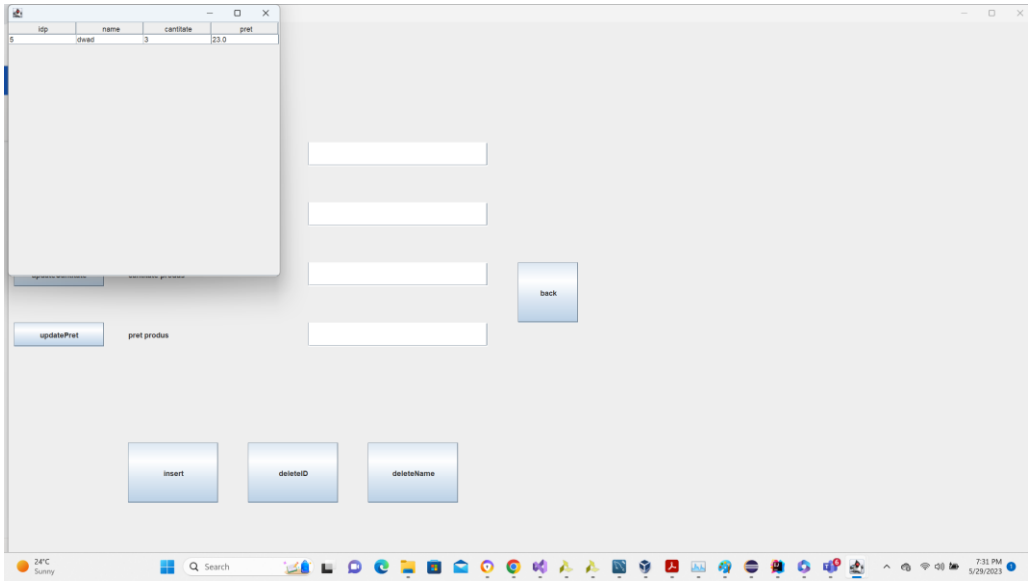
Clasele din presentation creeaza interfata grafica cu utilizatorul atat meniul principal cat si client produs si comanda

```java
public class MainFrame {

    public ClientFrame clientInterfata = new ClientFrame();
    public ProdusFrame produsInterfata=new ProdusFrame();
    public ComandaFrame comandaInterfata=new ComandaFrame();

    /**
     * creeaza frameul principal
     */
    public void createFrame() {
        JFrame mainFrame = new JFrame();
        mainFrame.setSize(1920, 1080);
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        mainFrame.setLayout(null);
        JLabel title = new JLabel("Order Management");
        title.setBounds(700, 200, 300, 100);
        title.setFont(new Font("Georgia", Font.BOLD + Font.ITALIC, 20));
        JButton client = new JButton("Client");
        client.setBounds(650, 400, 100, 100);
        JButton produs = new JButton("Produs");
        produs.setBounds(850, 400, 100, 100);
        JButton comanda = new JButton("Comanda");
        comanda.setBounds(750, 550, 100, 100);

        client.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                clientInterfata.createClientFrame();
                mainFrame.setVisible(false);
            }
        });

        produs.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                produsInterfata.createProdusFrame();
                mainFrame.setVisible(false);
            }
        });

        comanda.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

5.REZULTATE


idc
name
address
email
age
1
Gica
planete
planete@utcl...
38
2
edwdw
dewde
adead
25
7
Cristinuta
garli
cristinuta@utc...
20
23
adawdw
adwawd
wadwad
45
27
cAsedcaasd
vdardavdvds
rvaddavdf
38
78
CSCS
cCCAS
cesae
90
back
updateEmail
email client
updateAge
varsta client
insert
deleteID
deleteName
24°C
Sunny
Search
7:32 PM
5/29/2023


idco
numeClient
numeProdus
cantitate
1 Ion slapi 5
2 Gica dwad 2
3 Ion slapi 5
6 Cristinuta dwad 40
20 Ion slapi 5
24 Gica dwad 1
generateBill
createCommand
refresh


idp
name
cantitate
pret
5 dwad 3 23.0
back
updatePret
pret produs
insert
deleteID
deleteName

6.CONCLUZII

Tema presupune dezvoltarea cunostintelor si a abilitatilor de a realiza operatii asupra unei baze de date(insert, update si delete) folosind java .

7.BIBLIOGRAFIE

• Connect to MySql from a Java application o https://www.baeldung.com/java-jdbc o http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

• JAVADOC o https://www.baeldung.com/javadoc